EMU: Efficient Muscle Simulation in Deformation Space

Yona Uzan
Ecole Normale Supérieure Paris-Saclay
France
yona.uz@gmail.com

Gauthier Multari
Ecole Normale Supérieure Paris-Saclay
France
gauthier.multari@ens-paris-saclay.fr

Abstract

This report presents our work regarding the paper EMU: Efficient Muscle Simulation in Deformation Space [16]. It was done in order to validate the course of Geometric Data Analysis of Mr. Feydy of the Master MVA (Mathematics, Vision, Learning) at ENS Paris-Saclay. We chose to work on a subject of Space and Form applied to biomechanics. This paper presents a new way of simulating muscle driven skeletal motion. Compared to the state of the art Finite Element method simulation, this method allows for a hollistic approach to musculoskeletal animation while being more computationnaly efficient than previous state-of-the-art methods.

1 Introduction

Muscle simulation is a long time problematic of computer graphics. Having realistic reproduction of the behaviour of muscle driven skeletal movements has many applications. We can cite animation and VFX, where realistic muscle-driven skeletal animations are crucial for creating lifelike characters. They allow for more natural and nuanced movements, enhancing the overall visual quality of films, TV shows, and video games [11]. In medical training and education, realistic muscle-driven skeletal simulations provide a safe and effective environment for medical professionals to practice minimally invasive procedures, surgical techniques, and patient counseling. For instance, Sun et al. developed a dynamic muscle-driven simulation of laparoscopic surgical procedures based on 3D patient-specific anatomy. Realistic muscle-driven skeletal movements can also offer insights into athlete bio-mechanics, performance optimization, and injury prevention in sports analysis and training. Wang et al. developed a muscle-driven model for motion analysis and injury risk prediction of football players.

There are multiple methods for musculoskeletal simulations. The first is coarse mesh simulation, that heavily simplifies the geometry and the materials properties of the models. This simplification leads to a faster computation time than other methods at the expense of accuracy and variability of use cases [14]. A second method is projective dynamics. It involves mapping a complex system onto a simpler, more manageable representation but such representations limits the applicability of certain types of material models and can lead to unrealistic behaviours.

The presented work is based on quasi-static simulation methods. These methods are particularly suitable for systems where the time scales of interest are much larger than the characteristic dynamic response times of the system. In other words, quasi-static simulations assume that the system is in quasi-equilibrium throughout the simulation process. Quasi-static simulations assume that the system is always close to an equilibrium state during the simulation. This implies that the system has sufficient time to adjust to changes in its configuration or loading conditions. Instead of directly solving dynamic equations of motion, quasi-static methods perform time-incremental analyses. The simulation proceeds in small steps, and at each step, the system is assumed to be in quasi-equilibrium. The governing equations are then solved incrementally to update the system's state. Quasi-static simulations typically assume that deformations and rotations of the system are small. This assumption simplifies the analysis and allows for the use of linearized constitutive models. In each time increment, the stiffness matrix of the system is updated based on the current configuration. This accounts for changes in geometry, material properties, or boundary conditions.

Based on this principle, the paper presents a method which allows for a high resolution simulation of muscles, tendons, bones and joints while having better computational performance than other visually accurate methods.

2 Related works

The most popular methods for bulk musculosckeletal simulation (the simulation of multiple bones and muscles together) are Finite Element Methods (FEM). The first step of FEM is to divide the physical structure or system into smaller, simpler shapes called elements. These elements are connected at points called nodes. Nodes are the points where the physical properties of the system are evaluated, and elements are the interconnected subdomains that approximate the behavior of the system. Functions called shape functions are used to interpolate the physical quantities within an element based on the values at its nodes. These functions ensure a smooth representation of the variable of interest across the element.

The Finite Element Method is often derived from a variational principle, where the solution minimizes the total

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
2018. ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

potential energy or work done in the system. This leads to a set of partial differential equations. The problem is decomposed in the following elements:

- The Stiffness Matrix Represents the resistance of an element to deformation. It is derived from the material properties and geometry of the element.
- The Mass Matrix represents the mass distribution within an element.
- The Damping Matrix represents the damping characteristics of the material.
- The load Vector represents external forces applied to the system.

The system-level equations are assembled by combining the contributions from all the elements in the system. This involves arranging the element matrices and vectors into a global system. The resulting system of linear equations is solved using numerical techniques such as direct solvers (Gaussian elimination) or iterative solvers (conjugate gradient method).The results are then post-processed to obtain the desired quantities and visualize the behavior of the system.

The FEM method is very computationnaly intensive, often leads to numerical stiffening on heteregenous material, and is difficult to parallelize. Most of recent research mainly focused on improving the speed of the method in this complex context of bulk musculoskeletal simulation.

One proposed method is to use a multigrid method [5]. The key idea behind multigrid methods is to combine the efficiency of coarse grids (representing low-frequency components of the solution) with the accuracy of fine grids (capturing high-frequency components) to accelerate the convergence of iterative solvers. The issue is that muscle simulations require several factors such as heterogeneous nonlinear material, complex geometry, and time-varying activation which makes the construction of effective multigrid hierarchies very difficult.

Another approach involves describing the musculoskeletal system through line-of-force models. In this representation, each muscle is conceptualized not as a volumetric structure but as a line—a three-dimensional curve that may incorporate wrapping surfaces or via points. Along these lines, contractile forces can act, while the skeletal system is depicted as a collection of interconnected rigid bodies [24]. Unlike volumetric models, lines of force do not account for volumetric deformations or capture the intricate details of muscle fiber configurations. To address this limitation, coupling with rigid skeletons becomes necessary [22].

Reducing the resolution of the simulation mesh can also lead to faster computations, as highlighted in [9]. However, this acceleration comes at the expense of accurate deformations. Notably, when employing such coarser meshes, muscle fiber fields either need to be disregarded or adjusted through experimental methods. Recently, rapid projective techniques have been employed in muscle simulation [13], those techniques involve coarsening the simulation mesh and imposing limitations on the applicable constitutive models. While improving computation times, this often leads to animations exhibiting stiffness and artifacts [20].

Eulerian methods, as demonstrated in [8], can also be used. Those methods utilize a fixed grid in the computational domain. Unlike Lagrangian methods that follow the motion of material points, Eulerian methods track the evolution of physical properties at fixed points in space over time. Nevertheless, their limitation lies in the simplification of muscle representations, as they do not explicitly account for the intricate details of muscle fiber configurations. It is noteworthy that these methods typically forego the explicit modeling of muscle fiber fields. Instead, they heavily depend on kinematic coupling to rigidly simulated bones to approximate muscle behavior.

Additionally, data-driven approaches have gained popularity in the broader context of biomechanical modeling [19]. However these techniques, while effective for modeling the entire body, generally do not incorporate explicit muscle modeling, focusing more on global body dynamics.

The primary focus of the EMU is on the development of an unreduced and efficient algorithm for conducting muscle-first simulations in musculoskeletal systems. The key factor contributing to the success of our algorithm, EMU, lies in its utilization of deformation gradients as the degrees-of-freedom for the simulation, diverging from the conventional use of nodal positions. This approach bears resemblance to discontinuous methods employed in shape modeling [3]. However, the challenge of seamlessly stitching together a continuous mesh from discontinuous elements remains an open problem.

An alternative discontinuous strategy involves the use of rotation-strain coordinates. Unfortunately, this approach can yield results that significantly differ from the gold-standard finite element approach [18]. EMU sets itself apart from previous discontinuous methods by measuring discontinuity through the explicit minimizer of a coupling energy, rather than simultaneous minimization of the coupling energy and physics energy. In this aspect, EMU shares common ground with projective dynamics [4] or Fast Automatic Skinning Transforms. However, distinctions exist between EMU and projective dynamics, mainly the fact that it can simulate muscle driven motion, anisitropic fibers as well as tendons without coupling terms.

3 Method developed in EMU

The EMU approach consists in representing the muscle as a 3D curve line and the skeletal system as a set of articulated rigid bodies. The musculoskeletal system relies on forces which apply on the system. Tendons are simulated as three orders of magnitude stiffer than muscles. No coupling term is introduced to handle joints and bones because hinge joint is represented as a shared edge between the two bone regions.

The overall movement of a musculoskeletal system is simulated using a quasi-static elasticity approach, where the driving force is the varying activation of muscle groups. In this study, motions dominated by dynamics, such as leaping, running, or punching are excluded. Focus is done on the muscle-induced deformation of the musculoskeletal system, disregarding inertial effects. The study is mainly focused where muscle activation exhibit slow acceleration, and albeit non-trivial.

An efficient solver for muscles In the EMU model, the quasi static deformation at a time t is discretized to be numerically resolved. It is the solution of Equation 1

$$\arg\min_q \int_\Omega \left(\Psi_{\text{iso}}(F(q)) + \Psi_{\text{fiber}}(F(q), u, a(t)) - W(q)\right) \, dQ \tag{1}$$

where Ψ_{iso} is a Neo-Hookean isotropic elastic potential, commonly used for hyperelastic models. Then, the strain energy density function for Neo-Hookean materials is given by:

$$\Psi_{\text{iso}} = \frac{\mu}{2}(I_1 - 3) + \frac{\lambda}{2}(J-1)^2, \tag{2}$$

F is the deformation gradient, q the deformed positions of corresponding rest positions Q over the domain Ω. An linear activation along a piecewise-constant direction field is used to define Ψ_{fiber}:

$$\Psi_{\text{fiber}}(F, u, a(t)) = a(t)uTF^TFu, \tag{3}$$

a(t) represent the non-negative activation, u is the unit-length fiber direction vector at the each tetrahedron.

The system is submitted to constraints such as pinning points, fixing regions as entire bones or constraining neighboring bones to rotate according to a specified joint. Moreover, bones are seen as rigid bodies whereas muscles are seen as softer bodies. The discretization step is done through the introduction of independent variables F_i representing the deformation gradient for each of the m tetrahedrons $i \in (1...m)$.

As the interest is mainly to represent continuous deformation, for each deformation gradient F_i, we aim to fin the the nearest continuous mesh ie the optimal deformed vertex positions q*. The goal is to satisfy the next equation in a least square sense.

$$E_C(q, F) = \frac{1}{2}q^TG^TGq - q^TG^TF + \frac{1}{2}F^TF \tag{4}$$

G is the sparse matrix that computes the actual deformation gradients from the mesh deformed according to q. Then the optimal solution is found as,

$$q* = \arg\min_q E_C(q, F) = (G^TG)^{-1}G^TF \tag{5}$$

Now, the minimization of 1 with less variables becomes the minimization problem over F:

$$\min_q E(F) \tag{6}$$

where

$$E(F) = \Psi_{\text{iso}} + \Psi_{\text{fiber}}(F, u, a(t)) + \alpha(E_C(q(F), F)) - W(q(F)) \tag{7}$$

and α is a a scalar parameter controlling the continuity implied by F. The probleme is simpler to be computed, it requires computing the e gradient of the objective function as $\frac{dE}{dF}$. EMU resolves this with a quasi Newton approach. We assuming that a such approximation can be done and experimentally reflects 90% of the matrix

$$(G^TG)^{-1} \approx \Phi^T\Lambda^{-1}\Phi. \tag{8}$$

Modi et al. solve this with the Hessian, following these steps:

$$\frac{d^2E}{dF^2} \approx H - \alpha B^T\Lambda^{-1}B \tag{9}$$

with $H = \frac{\partial^2\Psi_{\text{iso}}}{\partial F^2} + \frac{\partial^2\Psi_{\text{fiber}}}{\partial F^2} + \alpha I$ and $B = \Phi G^T$.

James and Pai introduce the resolution with the Woodbury matrix identity, that leads to :

$$\left(\frac{d^2E}{dF^2}\right)^{-1} \approx H^{-1} + \alpha H^{-1}B\left(\Lambda - \alpha BH^{-1}B\right)^{-1}BH^{-1} \tag{10}$$

Then, Modi et al. solve Equation 6 by successive quasi Newton iterations and a back-tracking line search, satisfying the Armijo condition.

Affine bones simulation The model is constituted of muscles, bones and tendons. The muscle solver has just been presented in the previous section. The rigidity of the bones make them deform much fewer than muscles. This criterion allows us to reuse the muscle model by choosing the F_i as a single strain gradient. So a single affine transform for each bone, that is represented as affine constraint in Equation 11 as $Jq = 0$. Moreover a high young's modulus characterize bones. Indeed young's modulus is a numerical constant that describes the elastic properties of a solid undergoing tension or compression in only one direction, that is the case for bones. Young's modulus is equal to the longitudinal stress divided by the strain, it is a measure of the ability of a material to withstand changes in length when under lengthwise tension or compression.

$$\begin{cases} q* = \arg\min_q E_C(q, F) = (G^TG)^{-1}G^TF \\ \text{subject to} B_q q = B_F F \\ Jq = 0 \end{cases} \tag{11}$$

The optimal deformed bones position is givent by Equation 11, where $B_f F$are the deformation gradients of the bones elements and $B_q q$ represents the deformation gradients of the bones on the continuous mesh. This yields to a linear system of the form $Ax = b$ as shown in Equation 13 which is solved by a parallelizable Pardiso solver.

$$\begin{pmatrix} G^T G & J^T & B^T \\ J & \epsilon_1 & 0 \\ B & 0 & \epsilon_2 \end{pmatrix} \begin{pmatrix} q \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} G^T F \\ 0 \\ B_F F \end{pmatrix} \tag{12}$$

Constraints and Modeling The system may be subjected to external forces as gravity which on the continuous tetrahedral meshes. In order to make the model converging, standard collision resolution is added into the algorithm. The quasi newton search algorithm is completed by collision detection between muscles and bones. In fact, those constraints are not really biologically feasible because of fascia, which are sheaths of connective tissue that limit deformation and contact. The musculoskeletal model is firstly manually set by labelling tetrahedrons as muscles or bones. Tendons are selected among muscles near the origin. The fiber direction u is computed such as the system respects the heat equation $\Delta f = 0$. The α value introduced in Equation 9 is an energy weighting term which makes the EMU deformation closer to FEM when it increases. Moreover, the higher α increases the stiffness of the system and the number of newton iterations to convergence.

4 Results of EMU simulation

The original paper presents various experiments with different sizes of mesh. The mesh sizes range from 3k tetrahedron up to 600k. The first stage is about identifying the initial k modes of the Hessian (as per Equation 10) in a single upfront computation. The duration of this pre-processing phase varies notably depending on the mesh connectivity and the number of bones in the mesh, spanning from several seconds to several minutes in the case of larger examples. The subsequent pre-processing step entails the selection of the parameter α involving a maximum of 10 Newton solves on the mesh. As these are one-time operations, they were excluded from the performance metrics.

Superior performance is demonstrated by EMU when compared to the state-of-the-art open-source FEM solver at the time of the release of the paper [12]. The tests were done using the Stable Neo-Hookean elasticity simulation from [20], solved with the open-source Pardiso solver [6]. Evaluation of both algorithms is based on scalability concerning mesh size and the number of available CPU cores. Convergence testing involves monitoring the change in energy during an iteration, ensuring it is < 1e-2 for reliable results with excellent visual fidelity. The scaling tests focused on the simple fusiform muscle, and all meshes, including the 600k tetrahedral mesh, ran on a 16GB RAM system without encountering memory limitation issues. This is due to the fact that dense matrices are fixed-size, so memory usage poses no concerns.

On a single-core CPU, EMU demonstrates superior scalability compared to the state-of-the-art Finite Element Method when the amount of tetrahedras is above 50k, the difference between the two mehtods only increases as the amount of tetrahedras does (Fig.1).

The key computational operations, involving sparse back substitution and dense matrix inversion, are efficiently handled by EMU, contributing to its enhanced performance. Increasing mesh resolution has a minimal impact on the spectral characteristics of the Hessian, resulting in a consistent cost for the required dense solve. As a result, EMU outperforms FEM across various examples, especially for medium to large meshes, showcasing impressive speedups exceeding 20 times. Remarkably, EMU achieves this acceleration without reducing the solution space, solving the same problem as FEM discretization. Even though not explicitly designed for isotropic, homogeneous materials, EMU maintains its performance advantage over FEM, as it demonstrated by a 5-6x speed-up on the Stanford Bunny mesh. EMU exhibits good parallelization performances aswell, with up to an additional 3x performance improvement when executed on a 12-core machine with hyperthreading disabled. The parallelizability of the EMU Hessian update is highlighted, but the current bottleneck lies in the linear solver (Pardiso), which faces challenges in efficiently parallelizing the backsolves essential for minimizing the ACAP energy.

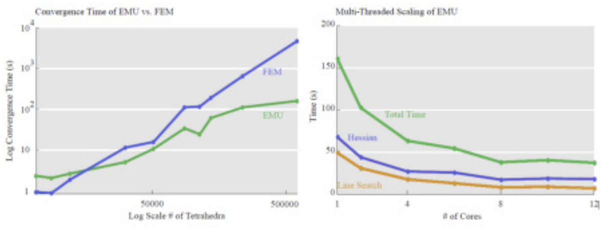

Figure 1. Left: Convergence time of EMU vs FEM in function of the number of tetrahedra on a single core CPU. **Right:** Convergence time of EMU vs FEM in function of the number of cores on a multi core CPU

The EMU deformation gradient formalism offers a notable advantage in its seamless integration of joints. Illustrated in Figure 1 is the reciprocal interaction between the contraction of the bicep and the resulting motion of the humerus, radius, and ulna. The biarticular nature of the bicep, establishing a direct connection between the shoulder and forearm, imparts motion to the humerus as the muscle drives the forearm. EMU adeptly accommodates intricate muscular structures, effectively transmitting forces through both the elbow (modeled as a hinge joint) and the shoulder (modeled as a spherical joint). As depicted in Figure 3, a simulation featuring a contracted hamstring demonstrates EMU's capability to generate realistic, muscle-centric motion in complex scenarios involving multiple muscles, joints, tendons, and bones.

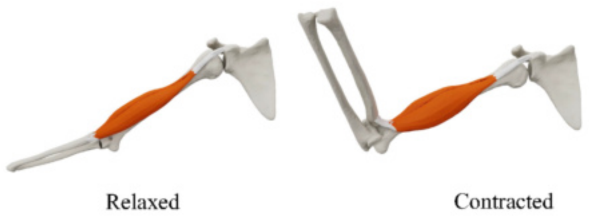

Figure 2. Simulation the motion of the humerus, radius and ulna induced by contracting the bicep.

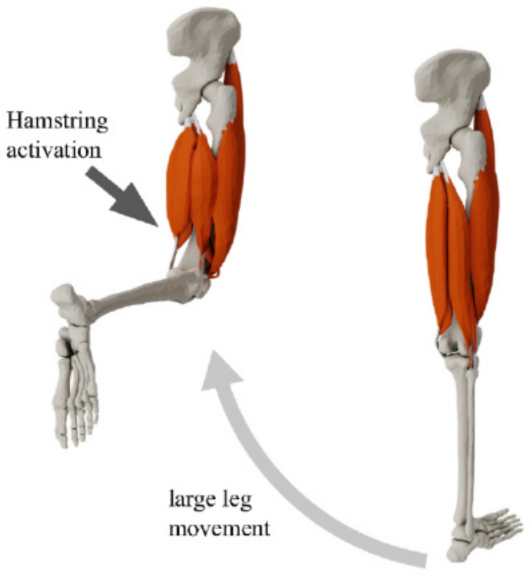

Figure 3. Simultation of the motion of a leg induced by the contraction of the hamstring

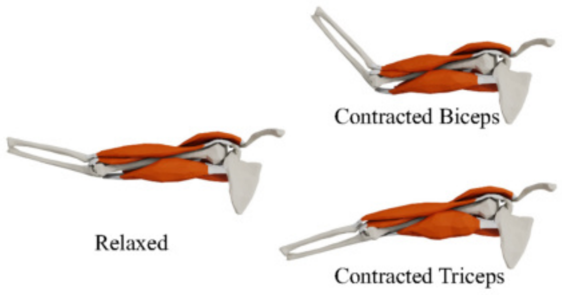

Figure 4. Simultation of the motion of the complex interaction of both biceps and triceps in a realistic manner

The method is also able to simulate more complex biomechanical models, figure 4 highlights EMU's capability to generate simulations with realistic biomechanical activation sequences. The depicted scenario involves a bicep contraction succeeded by a tricep contraction, resulting in the flexion and subsequent hyperextension of the elbow. This comprehensive example underscores EMU's efficiency and its seamless animation of bones, tendons, and muscles to generate muscle-first bulk musculoskeletal motion.

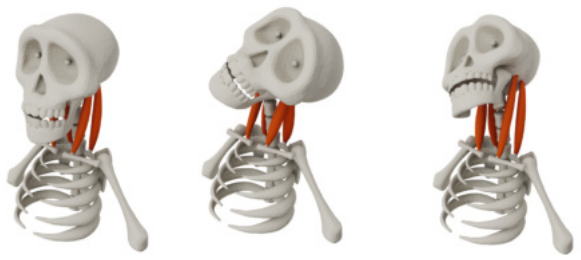

Figure 5. Simultation of a head roll by hollistically simulating the neck joins

Finally, figure 5 demonstrates EMU's utility in generating muscle-first head motion, where a cartoon head roll is exclusively driven by muscle actuations of the four neck muscles.

5 Experiments

It is important to be noted that none of the members of the groups had any training or experience regarding finite elements simulations. We started our experiments with the Mesh generation. Indeed, in order to visualize the deformation involved, we have to apply the method to a 3D model . This is a preliminary step that was not developed in the paper but that is essential to simulate muscle deformation in space and time. The muscle has a complex geometry as it acts at different scales: spatial and temporal. It is subject to forces as previously seen and respects constraints. We inspired from Maier and Schulte's work : *Mesh generation and multi-scale simulation of a contracting muscle–tendon complex*. In the paper [15], they generate a spline approximation of the muscle surface from a point cloud. That point cloud stems from the segmentation of medical imaging data. To further enhance the mesh quality of each slice, they apply a Laplacian smoothing. Some quadrangulations are then generated and elements on neighboring are connected.

Creating 3D mesh files is a complex procedure. It is possible to generate very simplistic meshes consisting of a small number of triangles in Python using the Poly3DCollection library. The figure is the first generation of mesh very simplified using this library. The coordinates of the triangles are given manually, which makes the creation of mesh very long in Python cf. Figure 6.

We used MeshIO Python library to work with various mesh file formats. It allows users to read and write meshes in different formats, making it a valuable tool for mesh-related tasks in scientific computing, finite element analysis, and

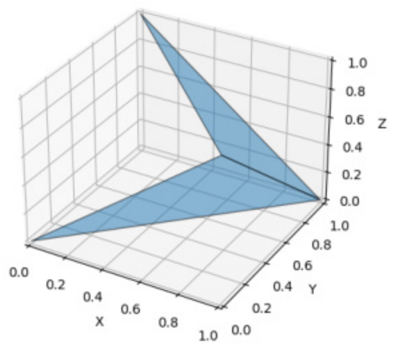

Figure 6. 3D mesh manually generated with Poly3DCollection in Python

computational geometry. Figure 7 is the plot of the mesh file. We designed a matrix to deform it, the right image is its deformation through the matrix gradient F. We considered q the first mesh file and F matrix of deformation gradient. The dot product of them gave the second mesh file.

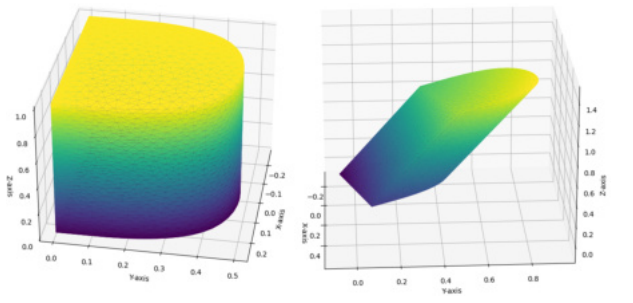

Figure 7. Left A simple mesh generated in Python and **Right** its deformation through deformation matrix F, which is here equal to

$$F = \begin{pmatrix} 1.2 & 0.1 & 0.2 \\ 0.3 & 0.9 & 0.4 \\ 0.5 & 0.6 & 1.1 \end{pmatrix} \tag{13}$$

We then tried to generate a mesh closer to an actual muscle. We opted for a ellipse (Figure 8).

Once more familiar with 3D mesh manipulation we tried to implement the method in python: solving the equations, subjecting the systems to forces, constraints. Our trials were inconclusive, and we did not manage to get the code to work. The code can be found here : [2] Being very time limited for the project, we looked for open source tools that could help us. After various experiments we decided to focus on Fenics cf. [1].

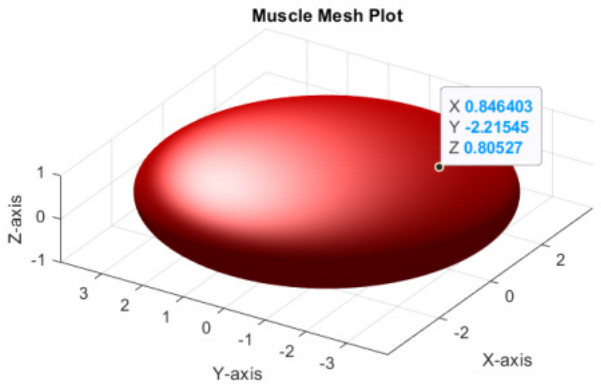

Figure 8. 3D muscle simulated as an ellipse

Fenics is an open source scientific computing platform designed for the automatic resolution of mathematical problems involving partial differential equations. It deploys a powerful computing capacity and allows to generate 3D objects from mesh. Designed especially for researchers and engineers, it allows to focus on the formulation of the problem rather than the details of the digital implementation. It allows the resolution of a number of equations like Poisson, Dirichlet, heat equation, Neumann and resolution in a finite element analysis.

It is a fairly recent platform available on Linux and Mac. We encountered multiple installation problems which slowed our results. Indeed, Fenics uses related libraries like Dolfin useful for problem solving. Installing the tools on Ubuntu and on Conda via a Windows PC is not optimal. We will then present the equations that were useful to us in solving our problem although this did not work.

Knowing that we have available an initial deformation gradient conjecture F, and an initial position matrix, the configuration of the finite element problem consists in defining the boundary conditions, the integration measures on the domain and limits and the hardware model that is the Neo-Hookean model as seen in Equation 1. We mainly chose to work on deformation gradient formalism more than the quasi-static elasticity approach. We tried first working on the simulation of muscles before extending the model to bones and tendons. We tried to program some functions like the Gradient calculation, the Neo-hookean and the Woodbury method. Our trials and codes are available here [2].

6 Limitations and extensions

One the main limitations of this paper's method is that soft tissue behavior is regarded as quasi-static at each time step and it is therefore unable to fully couple the inertial and large deformation effects.

Tendon representation is relatively basic and may not accurately capture the behavior of tendons under different loading conditions. Developing more sophisticated tendon models that can capture the non-linear behavior of tendons under different loading conditions. This could involve using more complex material models or incorporating strain energy potential functions.

The model's energy conservation properties have not been rigorously tested, and it may not accurately conserve energy over long simulation times. Furthermore, the model does not consider the effects of thermomechanical coupling, which could be important for simulations involving muscle contraction and heat production.

On a more practical side, the manual authoring system for constructing muscle and bone structures can be time-consuming and error-prone, especially for complex models. Developing parametric muscle models that can be easily customized to represent different muscle types and structures. This could allow for more rapid prototyping of new muscle models and reduce the need for extensive manual parameter tuning.

Another way to get faster nonlinear time integration could be by combining the frozen factorization approach [7] and the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm [17]. With the frozen factorization approach, the Hessian matrix is kept constant during the nonlinear solve, which can be advantageous since it is often expensive to form and factor the Hessian.

7 Conclusion

The original paper presented a new algorithm for bulk muscle simulation. It uses a new approach to the problem, a minimal energy penalty, which can be compared to a loss minimization problem in Machine Learning. This approach leads to a better performance scaling with high numbers of tetrahedras than the others state of the art FEM methods. It also proposes a new hollistic approach incorporating both and tendons to the model that does not use additional methods such as coupled rigid body simulations. The increase of speed of the method leads to lower accuracy simulations, while this is not noticeable visually in the examples given in the paper, this method can not be used as is for uses that need high accuracy results.

We unfortunately could not produce substantial results based this paper, but this project introduced us to finite elements methods and their applications. Given that both of us plan to work on computational solutions for health, this subject was of great interest to us.

References

[1] [n. d.]. Fenics. https://fenicsproject.org/.

[2] [n. d.]. Git. https://github.com/GauthierMlt/GDA-MVA.

[3] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. 2006. Primo: Coupled prisms for intuitive surface modeling. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. Eurographics Association, Aire-la-Ville, Switzerland, 11–20.

[4] Sofien Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. 2014. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Transactions on Graphics* 33, 4 (July 2014), 154:1–154:11. https://doi.org/10.1145/2601097.2601116

[5] A. Brandt. 1977. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.* 31, 138 (1977), 333–390.

[6] Arne De Coninck, Bernard De Baets, Dimitrios Kourounis, Florian Verbosio, Olga Schenk, Steffen Maenhout, and Jan Fostier. 2015. Needles: Toward large-scale genomic prediction with marker-by-environment interaction. *Genetics* 203, 1 (2015), 543–555. https://doi.org/10.1534/genetics.115.179887 arXiv:http://www.genetics.org/content/203/1/543.full.pdf

[7] Peter Deuflhard. 2011. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. Vol. 35. Springer Science & Business Media.

[8] Y. Fan, J. Litven, and D. K. Pai. 2014. Active volumetric musculoskeletal systems. *ACM Transactions on Graphics* 33, 4 (July 2014), 152:1–152:9. https://doi.org/10.1145/2601097.2601215

[9] Alexandru-Eugen Ichim, Petr Kadleček, Ladislav Kavan, and Mark Pauly. 2017. Phace: Physics-based face modeling and animation. *ACM Transactions on Graphics* 36, 4 (July 2017), 153:1–153:14. https://doi.org/10.1145/3072959.3073664

[10] Doug L. James and Dinesh K. Pai. 1999. ArtDefo: Accurate Real Time Deformable Objects. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 65–72. https://doi.org/10.1145/311535.311542

[11] Jongwon Lee and Kyomin Park. 2015. Muscle-driven skeletal animation for dynamic human motion synthesis. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–12.

[12] D. Levin. 2019. Gauss. https://github.com/dilevin/GAUSS.

[13] D. I. Levin, B. Gilles, B. Madler, and D. K. Pai. 2011. Extracting skeletal muscle fiber fields from noisy diffusion tensor data. *Medical Image Analysis* 15, 3 (2011), 340–353. https://doi.org/10.1016/j.media.2011.01.005

[14] S. P. Magnusson, A. Thorstensson, and T. Olsson. 2004. A comparison of coarse and fine mesh simulations of muscle contraction. *Journal of biomechanics* 37, 5 (2004), 975–985.

[15] Benjamin Maier and Miriam Schulte. 2022. Mesh generation and multi-scale simulation of a contracting muscle–tendon complex. *Journal of Computational Science* 59 (2022), 101559. https://doi.org/10.1016/j.jocs.2022.101559

[16] V. Modi, L. Fulton, A. Jacobson, S. Sueda, and D.I.W. Levin. 2021. EMU: Efficient Muscle Simulation in Deformation Space. *Computer Graphics Forum* 40, 4 (2021), 1418–1429. https://doi.org/10.1111/cgf.14185

[17] Jorge Nocedal. 1980. Updating Quasi-Newton Matrices with Limited Storage. *Math. Comp.* 35, 151 (1980), 773–782. https://doi.org/10.1090/S0025-5718-1980-0572855-7

[18] Z. Pan, H. Bao, and J. Huang. 2015. Subspace dynamic simulation using rotation-strain coordinates. *ACM Transactions on Graphics* 34, 6 (Oct. 2015), 242:1–242:12. https://doi.org/10.1145/2816795.2818090

[19] Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J. Black. 2015. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics* 34, 4 (July 2015), 120:1–120:14. https://doi.org/10.1145/2766993

[20] B. Smith, F. D. Goes, and T. Kim. 2018. Stable neo-Hookean flesh simulation. *ACM Transactions on Graphics* 37, 2 (Mar. 2018), 12:1–12:15. https://doi.org/10.1145/3180491

[21] Min Sun, Jianming Yang, and Li Sun. 2020. Dynamic muscle-driven simulation of laparoscopic surgical procedures based on 3D patient-specific anatomy. *Medical Image Analysis* 68 (2020), 101629.

[22] J. Teran, S. Blemker, V. N. T. Hing, and R. Fedkiw. 2003. Finite volume methods for the simulation of skeletal muscle. In *Proceedings of the*

2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 68–74. http://dl.acm.org/citation.cfm?id=846276.846285

[23] Bin Wang, Shuai Wang, and Guowei Chen. 2021. A muscle-driven model for motion analysis and injury risk prediction of football players. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51, 11 (2021), 3289–3301.

[24] J. M. Wang, S. R. Hamner, S. L. Delp, and V. Koltun. 2012. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Transactions on Graphics* 31, 4 (July 2012), 25:1–25:11. https://doi.org/10.1145/2185520.2185521