Digital Object Identifier

MOOSE-based Finite Element Hyperelastic Modeling for Soft Robot Simulations

KEVIN WANDKE\textsuperscript{12} and Y Z\textsuperscript{123} (Member, IEEE)
\textsuperscript{1}Department of Electrical, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
\textsuperscript{2}Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
\textsuperscript{3}Department of Nuclear, Plasma, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
Corresponding author: Y Z (e-mail: zhyang@illinois.edu).

ABSTRACT While soft robotics is an emerging frontier of the robotics field, the accurate modeling of large deformations of soft hyperelastic materials remains a challenge. Herein, we build on the existing open-source Multiphysics Object Oriented Simulation Environment (MOOSE) to accurately model neo-Hookean hyperelastic materials under user-defined loads and large strains. Excellent agreement between the simulated and theoretical results is obtained for simple geometries. Next, we show that the application can accurately model and predict the response of a real, soft pneumatic actuator. To conclude, the advantages of the open-source nature of our simulation environment are shown by demonstrating how direct control over many simulation parameters can allow users to tailor the accuracy, convergence, and speed of their simulations.

INDEX TERMS Finite Element Analysis, Open Source Software, Soft Robotics, Solid Modeling, Systems Simulation

I. INTRODUCTION

In recent years, developments in the field of soft robotics have greatly enhanced the scope of work that robots can perform. As a result of their functionally infinite degrees of freedom, inherently compliant nature, and high power-to-weight ratios, soft robots are now seeing use in the areas of manipulation, human-robot interaction, and minimally invasive surgery [1]–[3]. Despite these recent advances, there remain several factors inhibiting wider adoption and use of soft robots. One such factor is the need for specialized software to perform modeling and simulation on these highly nonlinear multiphysics systems (often requiring solid mechanics, dynamics, contact, and fluid dynamics). Traditional robots can be readily modeled as rigid bodies using analytic methods, or numerically using efficient and widely available Computer Aided Design (CAD), Finite Element Method (FEM), and simulation software [4], [5]. These tools allow traditional robots to be designed and tested in virtual environments, greatly reducing the time, expertise, and cost required for development. In contrast, tools capable of simulating soft robots are often expensive, inaccurate, or have limited feature sets.

Some effort has been made to simulate soft robots using analytic constant-curvature methods [6], [7] or as hyper-redundant kinematic chains [8]. Other analytical models have been developed for specific purposes such as trajectory matching [9] However, despite being computationally efficient, these methods can only model simple loading conditions and are often inaccurate under large deformations or external loading. Others have used novel data-driven approaches to develop models of soft robots [10], these methods are powerful tools for analysis of systems, but these approaches are often black-box input output mappings, or domain specific (e.g. control). As a result, these approaches lack the utility of an accurate general purpose model.

More robust attempts to model soft robots make use of FEM software packages. Work on FEM in soft robotics often tries to predict and optimize the design of soft actuators, as seen in [11]. More specifically, work has already been done using COMSOL-Multiphysics to model the nonlinear nature of soft sensors [12]. ANSYS was used to model hyperelastic material responses and additional soft sensors [13] and explore and optimize a variety of actuator parameters [14]. ANSYS was also used to further characterize the responses of soft actuators, including frictional self-contact in [15]. ABAQUS has also been used to model the complex electromechanical behavior of hyperelastic soft sensors [16], to analyze multi-material fiber-reinforced pneumatic actuators.
Given these requirements, the MOOSE framework developed and actively maintained by INL was selected as the most suitable option for the base software. MOOSE is a modular FEM system making use of variable coupling for multiphysics simulation, and allowing users to create custom kernels, materials, and applications with ease [24].

MOOSE’s Tensor Mechanics module already contains some of the machinery required to run soft robots FEM simulations. As shown in Figure 1A, within this module strains are computed from displacement data, then multiplied by an elasticity tensor. This procedure produces a stress tensor, which can then be used to calculate displacements. The process is iterated until a solution within a user defined margin of error is obtained.

Within the original MOOSE algorithm, stress is calculated using linear elasticity, which linearly relates the stress and strain of a material with a static elasticity tensor. However, under large deformations like those routinely experienced by soft robots, linear stress-strain relationships no longer hold. To address this problem, materials undergoing large elastic deformations follow nonlinear constitutive models to describe the stress-strain relationships. These constitutive models are usually referred to as strain energy density functions and are usually written as a function of the first three invariants of the left Cauchy-Green deformation tensor (the square of local displacement due to deformation) [25], [26].

Figure 1B shows our implementation of these strain energy density functions in MOOSE. The gradient of the displacement variables is used to calculate a deformation gradient (a linear mapping between a line element in the unreformed configuration to a line element in the deformed configuration). The deformation gradient is then used to compute the left Cauchy-Green deformation tensor, which is in turn subsequently used to compute the stresses. This process is repeated iteratively until the simulation falls below a user-specified margin of error. This new architecture can circumvent the linear elasticity tensor, allowing stresses to be directly computed from displacements and nonlinear stress-strain behaviors to be modeled.

B. NEO-HOOKEAN MODEL

In this work, the simplest and most popular hyperelastic constitutive law, the neo-Hookean model, was implemented. The strain energy is modeled as $W = C_1(I_1 - 3)$ for incompressible materials [27], where $W$ is the strain energy density, $C_1$ is a material constant proportional the shear modulus $\mu$, and $I_1$ is the first invariant (or trace) of the left Cauchy-Green deformation tensor [28].

Taking the derivative of the strain energy density function with respect to the deformation gradient, multiplying by the transpose of the deformation gradient $F^T$ and then dividing by $J$ (the determinant of $F$ and a measure of the change in volume of a material element) yields the stresses on the material:

$$\sigma = \frac{1}{J} \frac{\partial W}{\partial F} F^T$$

(1)
For incompressible materials, this yields the following equation for the stresses [27]:

\[ \sigma = 2D_1(J - 1)I + 2C_1B \]  

(2)

where \( D_1 \) functions as a Lagrange multiplier used to enforce incompressibility, \( I \) is an identity matrix, and \( B \) is the left Cauchy-Green deformation tensor.

Eq. (2) was implemented into a custom material in MOOSE that overrode the default linear stress-strain constitutive equation. MOOSE was then directed to skip the calculation of the strain to improve computation time, as the strain could now be directly computed from the displacements and their respective gradients.

Thus, when a user supplies the constants \( C_1 \) and \( D_1 \), the model, built on top of the MOOSE framework will be able to compute the neo-Hookean response of a material with those material properties under large deformations. However, it should be noted that the model is not perfect, and as strains keep increasing our hyperelastic model fails to account for the potential changes in the material parameters under extremely large deformations [29]. Further work will likely include more sophisticated material models such as the Mooney-Rivlin or Ogden models [25], [26], [30], [31].

**III. SIMPLE LOADING CONDITIONS**

Throughout the validation procedure, we used material constants for Dragonskin 20, which could be obtained from a database of soft material properties [32]. Using the database, we found that the best value of \( C_1 \) for our simulations was 0.0822 MPA. This best value was obtained by computing the strain range of our experimental tests and using the database of soft material properties [32]. Using the database, we found that the best value of \( C_1 \) for our simulations was 0.0822 MPA. This best value was obtained by computing the strain range of our experimental tests and using the database of soft material properties [32].

Once the stress equation, Eq. (2) was implemented in MOOSE, it was important to test how well simulations produced by these equations matched the theoretical results. The first test that was performed was a comparison of the numerical load-strain curves generated by simulations to the theoretical ones. In both simulated and theoretical cases, the incompressible equations were implemented on a (100mm x 10mm x 10mm) beam undergoing uniaxial extension.

Prior to obtaining the numerical data, a theoretical stress-strain curve needed to be established. The first step was to rewrite Eq. (2) to give a stress-strain curve for uniaxial extension. Fortunately, these calculations have already been performed [27], [28], and it has been shown that a neo-Hookean material will undergo uniaxial displacement according to the following equation:

\[ \sigma = C_1(\lambda^2 - \frac{1}{\lambda}) \]  

(3)

Where \( \lambda \) is the stretch ratio, the deformed length divided by the original length. To convert this equation into a stress-strain curve, for every time step in the simulated results, a \( \lambda \) value was calculated. Subsequently, the pressure and the load can be calculated as shown in Figure 2A.

After obtaining theoretical results, the next step was to set up a simulation to evaluate the application’s accuracy. In order to obtain data for a wide range of stresses, a single transient solve was run where the external pressure created by the boundary conditions on the end of the sample increased linearly with time as seen in Figure 2B-D.
Figure 2A demonstrates the agreement between the theoretical and numerical results for the 1-D deformation of incompressible neo-Hookean materials. It also shows the results of our simulation using a linear elastic material as well. It does appear that the hyperelastic simulation slightly underestimates the load that should be required for a sample to reach a desired deformation. The simulation underestimates the stress by a maximum of 10.8 KPA with an average underestimation of 7.8 KPA. Additionally, we can see that the simulation of the linearly elastic material fails to converge at a much lower strain, and substantially underestimates the stress at higher strains.

**B. INCOMPRESSIBLE BIAXIAL EXTENSION**

After verifying that the incompressible neo-Hookean model implemented in MOOSE could accurately simulate uniaxial extensions, it is important to verify that the model was also accurate for more complex loading conditions. An obvious next step was an equibiaxial extension. In this test, a (100mm x 10mm x 100mm) flat plate was modeled to be fixed on two adjacent edges, with a pressure that increased linearly with time applied to the opposite edges. The sample was then pulled equally in both the x and z directions as shown in Figure 3B-C.

This specific loading condition was also selected because theoretical results are straightforward to derive. In this case, the theoretical result is:

$$\sigma = C_1 (\lambda^2 - \frac{1}{\lambda^2})$$  \hspace{1cm} (4)

The process described in the uniaxial extension section was repeated with the new theoretical equation, as well as a new mesh geometry and boundary condition. As shown in Figure 3A, in this test we again see that the simulation follows the theoretical curve quite well, obtaining a maximum underestimation of 7.0 KPA and an average error of 4.1 KPA. This test again indicates that our method is capable of accurately reproducing the theoretical results for compressible neo-Hookean materials under several loading configurations. However, the linearly elastic material again fails to converge at a lower maximum strain, and substantially overestimates the stress at larger strains.

**IV. SIMULATION AND TESTING OF A SOFT PNEUMATIC ACTUATOR**

After performing the previous tests and validations, it was established that the numerical method agreed with the theoretical results. Here, we will simulate a prototypical pneumatic actuator and compare against a real-world physical system.
One of the most commonly used components in soft robots is a cylindrical actuator [33]. These pneumatic actuators have four main advantages: radial symmetry, actuation in one direction, homogeneous material properties, and no reliance on contact mechanics for actuation. Taken together, these factors suggest that this type of actuator is a good representative system to test and does not require the implementation of a contact mechanics physics engine in the simulation.

As a reminder, both the simulated and experimental tests were performed with Dragonskin 20, with $C_1 = 0.0822 \text{MPa}$ and $D_1 = 8.22 \text{MPa}$. With these parameters, we were able to run a simulation with the internal cavity pressure increasing linearly with time, $p = t \times 0.0822 \times 0.5$ as shown in Figure 4B-D. This parameterization allowed for an internal pressure-displacement graph to be produced as shown in Figure 4A.

After this, the next step was to obtain experimental data from the real system. We designed and manufactured a physical system as shown in the diagram displayed Figure 5A and the photos shown in Figure 5B-D. The physical actuator was laid flat on a table and connected to a pressure measurement device as well as a pressure source. By opening a valve connected to the pressure source, the internal pressure of the system increased, and the actuator elongated. An image of the displacement was captured against a textured background, allowing for extension to be calculated as seen in Figure 5C-D. To mitigate friction, the bottom of the actuator was placed on friction-reducing pads.

This setup allowed for a pressure-elongation curve to be obtained for our experimental data. When the experimental data and the simulation data were plotted on the same chart, it was clear that there was agreement between the results at lower strains, but at higher strains the simulation underestimated the pressure (maximum error of 6.2 KPA, average error of 1.4 KPA). One possible explanation for this discrepancy is that the neo-Hookean hyperelastic model does not perfectly describe the material’s response at large deformations. Additionally, the simulation using linear elasticity was unable to converge to solutions at far lower strains than the hyperelastic model. Nevertheless, the model we implement in MOOSE is extremely useful to realistically simulate the deformation of soft robots.

After demonstrating the agreement between the simulated and real world experimental results, it was important to further improve the simulation platform. To do this, the open-source nature of MOOSE was leveraged. While other FEM packages such as ABAQUS, ANSYS, and COMSOL can all model hyperelastic materials, these packages are closed-source and limit the amount of direct control users have over the simulation environment. While the simplicity afforded by these packages is usually beneficial to users, in the case of more exotic and extreme simulations such as soft robotics, an open-source package offers several clear advantages. First, an open source software can be more easily integrated into other...
TABLE 1. Comparison of Simulation Parameters

| Test number | Key Parameter                        | Computation Time | Simulation Time (seconds) | Stretch (%) | Average Error (%) |
|-------------|--------------------------------------|------------------|--------------------------|-------------|-------------------|
| 1           | Baseline                             | 164.6            | 20.1                     | 394.4       | 0.18              |
| 2           | Preconditioner                       | No Convergence   | N/A                      | N/A         | N/A               |
| 3           | Preconditioner                       | No Convergence   | N/A                      | N/A         | N/A               |
| 4           | Preconditioner                       | 90.2             | 30.0                     | 503.4       | 0.13              |
| 5           | Preconditioner                       | 98.8             | 30.0                     | 503.4       | 0.13              |
| 6           | Preconditioner                       | 90.3             | 30.0                     | 503.4       | 0.13              |
| 7           | Preconditioner                       | No Convergence   | N/A                      | N/A         | N/A               |
| 8           | Preconditioner                       | 199.7            | 30.0                     | 503.4       | 0.13              |
| 9           | Preconditioner                       | 259.9            | 20.1                     | 394.4       | 0.14              |
| 10          | Preconditioner                       | 312.7            | 26.9                     | 471.6       | 0.13              |
| 11          | Timestepter                          | 164.6            | 20.1                     | 394.4       | 0.18              |
| 12          | Timestepter                          | 180.9            | 20.1                     | 394.4       | 0.16              |
| 13          | Timestepter                          | 253.5            | 20.1                     | 394.4       | 0.13              |
| 14          | Timestepter                          | 127.5            | 20.1                     | 394.4       | 0.16              |
| 15          | Timestepter                          | 146.8            | 20.1                     | 394.4       | 0.16              |
| 16          | TimeIntegrator                       | 156.1            | 30.0                     | 502.9       | 0.14              |
| 17          | TimeIntegrator                       | 717.4            | 16.2                     | 343.2       | 1.93              |
| 18          | TimeIntegrator                       | 279.3            | 21.0                     | 406.0       | 0.12              |
| 19          | TimeIntegrator                       | 193.3            | 20.1                     | 394.4       | 0.16              |
| 20          | TimeIntegrator                       | 152.5            | 30.0                     | 502.9       | 0.14              |
| 21          | TimeIntegrator                       | No Convergence   | N/A                      | N/A         | N/A               |
| 22          | Linear Iteration                     | 51.9             | 2.5                      | 88.6        | 0.65              |
| 23          | Linear Iteration                     | 349.6            | 28.8                     | 491.6       | 0.13              |
| 24          | Nonlinear Iteration                  | 241.2            | 8.3                      | 221.0       | 0.22              |
| 25          | Nonlinear Iteration                  | 164.1            | 21.8                     | 414.5       | 0.11              |
| 26          | Threads                               | 3045.0           | 29.8                     | 500.9       | 0.08              |
| 27          | Threads                               | 374.3            | 21.8                     | 414.6       | 0.14              |
| 28          | Threads                               | 179.9            | 20.1                     | 394.2       | 0.18              |
| 29          | Error Tolerance                      | 10× lower        | 8.1                      | 218.5       | 0.21              |
| 30          | Error Tolerance                      | 10× higher       | 20.2                     | 395.3       | 0.22              |
| 31          | Compressibility                      | TRUE             | 19.6                     | 406.9       | -3.96             |
| 32          | Volumetric penalty                   | No Convergence   | N/A                      | N/A         | N/A               |
| 33          | Volumetric penalty                   | 61.8             | 30.0                     | 503.4       | 1.57              |
| 34          | Rotations allowed                    | TRUE             | 15.6                     | 69.8        | 1.39              |
| 35          | No scaling                           | TRUE             | 44.4                     | 130.1       | 0.73              |
| 36          | Undisplaced Mesh                     | TRUE             | 29.4                     | 9.9         | 134.9             | 0.33

Simulation, control, or data analysis packages as users have the ability to directly control the types of inputs and outputs generated. Secondly, open source packages are usually free, making them attractive to a wide range of users who would be unable or unwilling to pay for their simulation software. Next, open source softwares allow users to directly control the simulation parameters, a useful quality for complex simulations such as those often performed in the soft robotics community. A final potential advantage over an open-source framework is that they can often be tailored to specific use cases, and thus can run far more efficiently than more general-purpose softwares.

While all of these factors contribute to the attractiveness of an open-source soft robotic simulation environment, in this section of the paper we intend to more thoroughly investigate the advantage that direct control over simulation parameters can afford end users. We intend to more fully explore both the advantages of integration with other software packages and well as potential gains in efficiency in future works.

The purpose of investigating a variety of simulation parameters is twofold. First, this section seeks to demonstrate that by using an open source software, users have enormous control over the simulation’s behavior, and can alter its performance to meet their needs. Second, this section will provide end users of our software with an understanding of how different simulation parameters can impact the convergence, accuracy, and computational cost of their simulations. Throughout the analysis five types of different simulation parameters are investigated. These parameters are: the preconditioner used in the Jacobian-Free-Newton-Krylov method, the time-stepping scheme, the time integration scheme, the solving parameters, and the model parameters. These dif-
Different parameters were selected based off of the significant impact that they had on the quality of the simulation. To assess the performance of changes to the simulation, three factors were considered: the time it took for the simulation to run (in seconds), how much strain the uniaxial rod could experience prior to the model failing to converge (percentage), and the average error between the simulated and theoretical results (percentage).

All tests were performed in a manner similar to the uniaxial extension test in section III on an Intel Xeon E5-2698 V4 2.2Ghz CPU. The maximum internal simulation time was set to 30 seconds, rather than 10, which corresponded to a strain of about 500%. Furthermore, the baseline test was run with the following parameters: Preconditioner=Hypre-BoomerAMG Timestepper=ConstantDT=0.1 Timeintegrator=ImplicitEuler, Linear Iteration=20, Nonlinear Iterations=10, Threads=10, Volumetric Penalty=0.2e8 Error Tolerance=1.0e-4, Scaling=Allowed, Rotations=Small, Mesh Displacement=True. The results can be found in Table 1 above.

From these results we can draw several useful conclusions. Firstly, we can see from Tests 2-10 that the choice of preconditioner has a dramatic impact on both the model’s convergence and speed. While the default preconditioner (Hypre-BoomerAMG) is recommended for most users, it can be seen that using a direct preconditioner (the LU family of preconditioners) allows for far better convergence as well as vastly improved speed.

Tests 11-15 show that implementing an adaptive time-stepping scheme can also reduce computational cost, and that a poor choice of initial time-step can increase the cost dramatically. Tests 16-21 show that time integration schemes that are supposed to be more accurate (4th order RK methods) are far more computationally costly and can have reduced accuracy in some cases. Furthermore, explicit time integrators can improve model convergence and speed without sacrificing accuracy.

Tests 22-25 show increasing the number of linear and nonlinear solver iterations for each time step will increase the cost, convergence, and accuracy of the simulation. The exception is increasing the number of nonlinear iterations. This reduces the number of failed time-steps, meaning that increasing this value can lead to decreases in computational cost. Tests 26-28 show that by increasing the number of threads that the process uses, computational time can be decreased. Interestingly, the parallelization is not perfect, and increasing the number of threads comes at a cost to model convergence and accuracy.

Tests 29 and 30 show that by increasing the maximum allowable error, the simulation can run much faster, without sacrificing much accuracy. Meanwhile stricter error tolerance will decrease model convergence. Test 31 shows that it is far better to use the incompressible version of the neo-Hookean equations as compressible versions performed worse in each performance metric. Tests 32 and 33 show that decreasing the volumetric penalty term can dramatically improve computation time and convergence, but at significant cost to accuracy. Lastly, tests 34-36 show that it is important to use the displaced mesh, automatic scaling, and small rotation parameters for simple tests where large displacements are to be expected.

In summary, the amount of control an open-source simulation platform gives its users allows them to tailor their simulations to closely fit their needs. This feature is not present in many other FEM software packages used for soft robotic simulation, giving this MOOSE-based system a unique advantage over them.

VI. CONCLUSION

In conclusion, this paper outlines a simulation method, implemented in MOOSE, for simulating soft robots using the neo-Hookean hyperelastic model. The ability to accurately predict and model the hyperelastic properties of soft materials within an open-source framework greatly enhances the ability of the soft robotics community to develop robots with specific performance characteristics in mind. Validation was performed by comparing the numerical results to the theoretical ones for both uniaxial and biaxial extension loading conditions. Next, MOOSE was used to simulate and compare with the deformation of a real-world soft pneumatic actuator. In all the tests, the model displayed good tracking of the desired result. Finally, a comparison of a wide range of simulation parameters was performed to show how the degree of control over the simulation offered by an open-source framework allows users to customize it to their own needs.

This work of developing an open-source modeling platform capable of predicting the motions of soft robots is vital to the continued development of the field. To date, no such open-source comprehensive platform exists that is capable of implementing hyperelastic multiphysics simulations. Our work will allow researchers to explore a large number of soft robotic designs and controls quickly, and to avoid the costly process of fabricating designs. Currently, without prior knowledge of how designs will interact with their environments, most work is done using a trial and error approach. Overall, this simulation platform will help mature and expand the technology of soft robots to expand the domains in which robots are used.

VII. CODE

The code for this project is available on Github at: https://github.com/Z-Laboratory/Kraken

ACKNOWLEDGEMENT

The authors like to thank David Null for his assistance in the design and fabrication of the physical prototypes as well as Alexander Lindsay, Benjamin Spencer, and Daniel Schwen for their assistance on getting started with MOOSE.

REFERENCES

[1] S. Kim, C. Laschi, and B. Trimmer, “Soft robotics: A bioinspired evolution in robotics,” Trends in Biotechnology,
Modelling and control of soft robots using the koopman operator and model predictive control,” CoRR, vol. abs/1902.02827, 2019. [Online]. Available: http://arxiv.org/abs/1902.02827

11. Y. Elsayed, A. Vincenzi, C. Lekakou, T. Geng, C. M. Saaji, T. Ranzani, M. Cianchetti, and A. Menciassi, “Finite element analysis and design optimization of a pneumatically actuating silicone module for robotic surgery applications,” Soft Robotics, vol. 5, no. 4, pp. 255–262, 2014. [Online]. Available: https://doi.org/10.1007/s40588-014-0016-2

12. M. Pozzi, C. Gaudeni, Z. Iqbal, D. Prattichizzo, and M. Malvezzi, “Modelling a sensorized soft layer for adding compliance to the environment in robotic manipulation,” in Advances in Italian Mechanism Science, V. Niola and A. Gasparotto, Eds. Cham: Springer International Publishing, 2021, pp. 370–377.

13. T. Bieze, F. Largillière, S. H. Chehade, M. S. Lopez, and T. Bieze, “Finite element modeling in the design process of 3D printed pneumatic soft actuators and sensors,” Soft Robotics, vol. 9, no. 3, 2022. [Online]. Available: https://www.mdpi.com/2218-6581/9/3/52

14. W. Hu, R. Mutlu, W. Li, and G. A. McGarvey, “A structural optimisation method for a soft pneumatic actuator,” Robotics, vol. 7, no. 2, 2018. [Online]. Available: https://www.mdpi.com/2221-6581/7/2/24

15. F. Yang, Q. Ruan, Y. Man, Z. Xie, H. Yue, B. Li, and R. Liu, “Design and optimize of a novel segmented soft pneumatic actuator,” IEEE Access, vol. 8, pp. 122304–122313, 2020.

16. M. Mooney, “A theory of large elastic deformation,” Journal of Applied Mechanics, vol. 1, pp. 113–125, 1948.

17. M. Hamza and H. Alwan, “Hyperelastic constitutive modeling of rubber and rubberlike materials under finite strain,” Eng. and Tech. Journal, vol. 28, no. 13, 2010.

18. L. Marechal, P. Balland, L. Lindenroth, P. Petrov, K. Kontovousois, and F. Bello, “Toward a common framework and database of materials for soft robotics,” Soft Robotics, pMID: 32589507. [Online]. Available: https://doi.org/10.1007/s40588-019-00115

19. T. Bieze, F. Largillière, S. H. Chehade, M. S. Lopez, and C. Duriez, “Fetch mark I manipulator,” [Online]. Available: https://softroboticstoolkit.com/fetch

KEVIN WANDKE received his B.S. in Mechanical Science and Engineering from the University of Illinois at Urbana-Champaign in 2019 and is currently pursuing his Ph.D. in Electrical and Computer Engineering at University of Illinois at Urbana-Champaign. He previously worked at Argonne National laboratory as a member of the SULI program, and at General Electric’s Global research center as an intern in the Edison Engineering Program. Additionally, he is the recipient of the Chancellor’s Scholarship and Olsen award for Excellence in Undergraduate Teaching.
Y Z is an associate professor and a Donald Biggar Willett Faculty Scholar in Department of Nuclear, Plasma, and Radiological Engineering, Department of Electrical and Computer Engineering, Program of Computational Science and Engineering, Center for Biophysics and Quantitative Biology, and Beckman Institute of Advanced Science and Technology at University of Illinois at Urbana-Champaign. He is also the Associate Head of Department of Nuclear, Plasma, and Radiological Engineering. He received his B.S. in Electrical Science and Technology from University of Science and Technology of China in 2004 and his Ph.D. in Nuclear Science and Engineering from Massachusetts Institute of Technology in 2010. He was a Clifford G. Shull Fellow at Oak Ridge National Laboratory from 2010 to 2012. YZ’s research can be summarized in three words: Molecules, Materials, and Machines. On the basic science side, his group synergistically combines and pushes the boundaries of theoretical and computational molecular science and neutron and X-ray scattering experiments. They develop accelerated molecular simulation methods, based on quantum and statistical mechanics and verified by experiments, which can effectively model a wide range of long timescale phenomena and rare events. The goal is to significantly extend our understanding of the equilibrium and non-equilibrium properties of natural and synthetic materials from the molecular and electronic level and provide accurate predictions and rational guidance to the design, synthesis, fabrication, and utilization of novel materials. Particular emphasis is given to the physics and chemistry of liquids, especially at interfaces, driven away from equilibrium, or under extreme conditions. On the applied research side, leveraging their expertise in novel materials and modeling, his group advances the development of soft robots and human-compatible machines, robots in extreme environments, intelligent control algorithms, and understandable artificial intelligence that can lead to immediate societal impact. He has been recognized with several awards including, most recently, the Dean’s Award for Excellence in Research and the American Nuclear Society Landis Award. He is an associate editor of Science and Technology of Advanced Materials.