AutoTMTDyn: A Matlab Software Package to Drive TMT Lagrange Dynamics of Series Rigid- and Continuum-link Mechanisms

S.M.H. Sadati[1,2,3], S. E. Naghibi[4], A. Shiva[2,3], S. Zschaler[2], H. Hauser[1], I. Walker[5], K. Althoefer[4] and T. Nanayakkara[3]

***Abstract*— Having a reliable accurate and at the same time simple dynamic model is important in analysis, design, path planning and control of robotic systems. Such models should be fast, convenient and simple to use to be accepted by the ever growing robotics research community. Among all the challenges, controlling complex systems with compound rigid and continuum body mechanisms with switching dynamical behavior, due to interaction with their environment in different applications, has attracted much research recently. To address these concerns, we have upgraded the AutoTMTDyn Matlab software package, a recently introduced tool for deriving Lagrange Equations of Motion (EOM) in a vector formalism, with the ability to handle series continuum-link mechanisms, e.g. continuum manipulators, and introducing an easy to use high level language (HLL) as a new text-based user interface. The variable curvature kinematics and TMT Lagrange dynamics of a continuum beam is derived in differential and discrete forms. The HLL elements are described and showcased in modeling and analysis of three bioinspired dynamical systems; lumped-system model of a spider web, continuum model for a rat whisker, and modeling a continuum pneumatic module. The challenges with using the package for systems with large modeling states, and the necessary modules to further automate the analysis of the dynamic systems are briefly discussed.**

Keywords- **Lagrange Dynamics, Continuum Mechanism, High Level Language, Software.**

I. INTRODUCTION

ROS, Orocos, SmartSoft, OpenRTM are some robotic software platforms developed to make robotics programming and configuration as accessible as possible to application domain experts. Domain-Specific Languages (DSLs) and Model-driven Engineering (MDE) are interesting emerging areas in the robotics research community, e.g. distributed robotics, system control, and vision, with the potential to facilitate the future programming of robots significantly. A DSL is a dedicated programming language for a particular problem domain, offering specific abstractions and notations, to decrease the coding complexity and increase programmer productivity. DSLs have been used for programming complex systems, e.g. robots, for which traditional general-purpose languages do not provide a good correlation between the implementation requirements and language features. To address this, DSLs are a powerful and systematic way to provide two main features; (i) quick and precise adaptation by domain experts, who are not familiar with general purpose programming languages; (ii) hiding the architecture complexity by software engineers to facilitate complex configuration and design architectures before transferring to domain experts. The High level Language (HLL) presented in this work is an initial effort in making dynamic modeling tools more accessible for interdisciplinary research in robotics.

On the other hand, modeling compound mechanisms with rigid and continuum links has been a challange in soft robotic research [1], [2], [3]. Newton-Euler, Lagrange, the Principle of Virtual Work (PVW) and TMT methods have been used to derive a dynamical system Equation Of Motion (EOM) [4], [5]. Based on the Newton-Euler method, the EOM of a system can be derived by knowing the relations between the system components. This method is straight forward but is complex to derive, results in large number of modeling states, and is not suitable for controller design. Matlb Robotics Toolbox is one of the software packages using this method [6]. Euler-Lagrange and Principle of Virtual Work methods use independent generalized coordinates to simplify the EOM derivation and support control design. Most of commercially available dynamical system modeling softwares, e.g. MSC. ADAMS, utilize Lagrange dynamics formulation [7]. However, the final set of equations can be complex and hard to interpret. Hence, an extra step is taken to collect Lagrange EOM in a close vector formalism. Alternatively, the TMT method is recently used by Wisse et al. for modeling passive biped walker [5]. The TMT method is based on parts of Lagrange's investigations on analytical mechanics, dealing with generalized coordinates, virtual work and inertial forces, which was published in in 1788 and before his well-known "Lagrange method". L. Schwab and M. Wisse called this method "TMT" because of the system inertia matrix in generalized coordinates ($T^{\mathrm{T}}MT$), where T is the Jacobean transformation matrix between the Cartesian and generalized coordinates spaces and M is the system inertia matrix. As a result, highest order derivatives are eliminated in the derivation process and the EOM are derived in a closed vector formalism. The

This work is supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC) Grant EP/N03211X/2, European Union H2020 project FourByThree code 637095, and Leverhulme Trust Project RPG-2016-345.

[1]S.M.H. Sadati and Helmut Hauser are with the Department of Engineering Mathematics, University of Bristol, Bristol BS8 1TH, U.K. (email: s.m.hadi.sadati@bristol.ac.uk)

[2]S.M.Hadi Sadati, Ali Shiva and Steffen Zschaler are the Department of Informstics, King's College London, London WC2R 2LS, U.K.

[3]S.M.Hadi Sadati, Ali Shiva and Thrishantha Nanayakkara are with the Dyson School of Design Engineering, Imperial College London, London SW7 2AZ, U.K.

[4]S. E. Naghibi and K. Althoefer is with the Department of Mechanical and Material Science, Queen Mary University of London, London E1 4NS, U.K.

[5]Ian D. Walker is with the Department of Electrical and Computer Engineering, Clemson University, Clemson, USA. (iwalker@g.clemson.edu)

derived terms are independent, hence suitable for parallel numerical implementation, and clear to interpret for control design. Changes in the system model due to change in the constraints or elements are easier to handle and do not require to fully re-derive the Lagrangian. Constraints, especially for parallel mechanisms and inverse dynamic investigation, are easy to implement too. "AutoTMTDyn" is a software package, developed to automate the derivation steps, based on Matlab programming Language and functionality, and with a text-based user interface [4]. We have used the package in modeling different dynamical systems recently [3], [8]. The application of the TMT method in modeling continuum manipulators is presented in [9]. In this research, a new HLL is proposed for the package for an easy-to-use and clear interface. The discrete derivation of a continuum dynamic system is investigated and implemented in the package.

In the following sections, first the terminology and elements of HLL used to design the text-based user interface is explained. Then the differential and discrete dynamics of a continuum beam as a series of continuum elements is explained based with the TMT method. Finally, three examples are discussed where the "AutoTMTDyn" package and the newly derived equations are employed for modeling bioinspired systems: a spider web, a rat whisker, and a pneumatic continuum manipulator. A short discussion is provided on the challenges in modeling large dimension systems and the necessary modules for further automation of the derivation and analysis of more general systems' EOM. The source codes used in this research are available at [10].

II. Materials and Methods

A. High Level Language for Text-based User Interface

Examples of the HLL proposed for the new text-based user interface is presented in Fig. 1-, 2-, and 3- middle. The user-interface are a Matlab code that needs to employ a HLL, to define the system geometry and properties, and calls "TMTEoM_QP", "EquilEoM", "SimEoM" and "AnimEOM" functions, to derive the system EOM, calculate the initial static equilibrium, simulate the dynamics, and animate the results respectively [10]. The HLL inputs are Matlab language structures that store properties of the simulation environment, named "wolrd", each body, named "body(i_b)", joints, "joint(i_j)", and external loads, "exload(i_l)", where i is a general numerator. "world.g" stores the gravity vector. "body(i_b).m" is the body mass, "body(i_b).I" is the inertia matrix, "body(i_b).l_ com" is the body Center of Mass (COM) and "body(i_b).tip" is the link tip which is used for an animation of the system in action. A rigid link is a single body with regular translational and rotational joints, but a continuum link a series of finite number of bodies interconnected with continuum joints. So we use "body(i_b)" for both body types. A "joint(i_j)" can be a regular joint between the system elements, spring/dampers, external inputs such as actuators, or constraints. "joint(i_j).first" and "joint(i_j).second" are the numbers associated with the first and second body that the joint is connected to. "joint(i_j).tr(i_t)" is the the joint i_t^{th} transformation matrix, there can be as many as needed, and has "joint(i_j).tr(i_t).trans" element for its Cartesian translational vector and "joint(i_j).tr(i_t).rot" for its rotational vector. "joint(i_j).tr(i_t).rot" can have two forms. It is a 1×2 vector with the first element for the axis of rotation and the second element for the amount of rotation. Alternatively, it is a 1×3 vector indicating that it is a continuum rotational link with transformation matrix based on curvatures/torsion of a beam. If any of the elements of "joint(i_j).tr(i_t).tr" or "joint(i_j).tr(i_t).rot" are free degrees of Freedom (DOF), they are indicated by "inf". For any DOF,there should be a "joint(i_j).dof(j_d)" element, where i_d is a numerator referring to the number of the associated DOF in the order that the joint DOFs are defined. "joint(i_j).dof(i_d).init" is the initial value, "joint(i_j).dof(i_d).input" is the input acting directly on the DOF, and if a spring/damper is needed parallel to the DOF, e.g. a revolute joint with a parallel spring/damper, "joint(i_j).dof(i_d).spring.coeff" is the spring coefficient, "joint(i_j).dof(i_d).spring.init" is the spring initial (reference) state, "joint(i_j).dof(i_d).spring.comp" is the spring compression ratio at pre-stress, and "joint(i_j).dof(i_d).damp.visc" is the viscous damping value. To impose symmetric constraints between the DOFs, "joint(i_j).dof(i_d).equal2" can set to the number of one of the previously defined DOFs to force an equality constraint. "joint(i_j).tr2nd" is the joint transformation matrix w.r.t. the second body with the same ".trans" and ".rot" elements. If the joint is a part of the mechanism original chain, i.e. the chain used to derive the system EOM and any alternative chain is considered as constraint, this can be left empty. "tr2nd" should be defined for the spring/dampers or constraints. In the case of a spring/damper, "joint(i_j).spring.coeff", "joint(i_j).spring.init", "joint(i_j).spring.comp", and "joint(i_j).damp.visc" are scalars for the spring coefficient, initial (reference) value, compression ratio (as initial pre-stress), and viscous damping respectively. The "joint(i_j)" is a constraint if "joint(i_j).tr2nd" is defined but none of ".spring" or ".damp" are defined. Only translational springs and dampers are supported at the moment. Finally, external loads are defined by "exload(i_l)". "exload(i_l).body" is the number of the body that the load is acting upon and "exload(i_l).tr(i_t)" is the transformation matrix w.r.t. the body local frame, with the same "trans" and "rot" elements as explained before.

B. Continuum Beam Variable Curvature Dynamics

A continuum beam with circular cross-section is considered, similar to the model presented for a bioinspired whisker setup in Fig. 2- left. External loads causes local strains ($v_{\hat{\xi}_i}$) and curvatures/torsion ($u_{\hat{\xi}_i}$) along a continuum beam, where $\hat{\xi}_i(s)$ is the i^{th} direction of a local curvilinear frame at the s axial location along the beam backbone. To derive the system differential mechanics, the beam can be considered as a finite number of infinitesimal beam elements with infinitesimal mass ($\mathrm{d}m = \sigma a \mathrm{d}s$), moment of inertia ($\mathrm{d}I = \mathrm{diag}[(3r_{\mathrm{w}}^2 + \delta s^2)/12,\ (3r_{\mathrm{w}}^2 + \delta s^2)/12,\ r_{\mathrm{w}}^2/2]\mathrm{d}m$), and linear

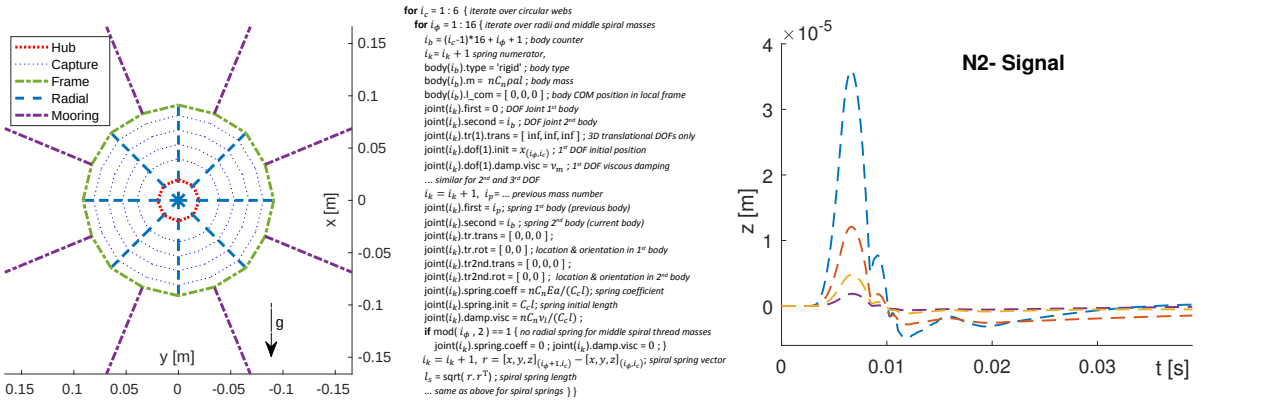

Fig. 1. Left- schematics of a spider web lumped system model; middle- parts of input code for driving EOM with "AutoTMTDyn" software package; right- sample simulation results for excitation on the left horizontal thread (N2).

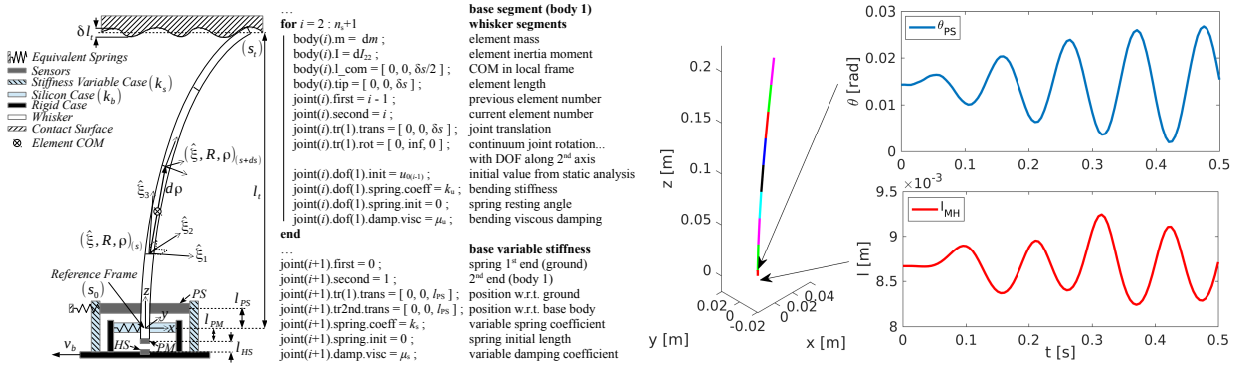

Fig. 2. Left- schematics of a bio-inspired whisker design and equivalent modeling framework; middle- parts of input code for driving EOM with "AutoTMTDyn" software package; right- sample simulation results for base element curvature (θ_{PS}) and translation (l_{HS}).

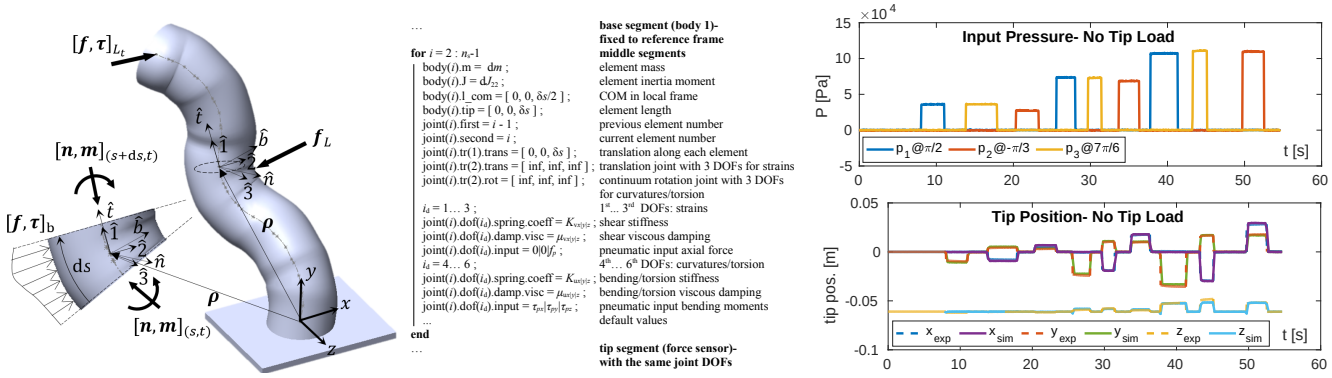

Fig. 3. Left- schematics of variable curvature kinematics and differential mechanics of a pneumatic continuum manipulator [9]; middle- parts of the input code for driving EOM with "AutoTMTDyn" software package; right- sample simulation results for a manipulator with three pressure inputs and no external load.

axial ($K_v = \mathrm{diag}[G\ G\ E]^{-1}a$) and bending/torsional ($K_u = \mathrm{diag}[EJ_{11}\ EJ_{22}\ GJ_{33}]^{-1}$) elasticities based on Hooke's law. E and $G \approx E/3$ are the beam elastic and shear modulus, σ is the beam material density, r_{w} and $a = \pi r_{\mathrm{w}}^2$ are the cross-section radius and area, J_{ii} is the second moment of area around the i^{th}-axis and $J = \mathrm{diag}[\pi r_{\mathrm{w}}^4/4,\ \pi r_{\mathrm{w}}^4/4,\ \pi r_{\mathrm{w}}^2/2]$.

The beam Variable Curvature (VC) kinematics is expressed by the following two differential equations for the Cartesian position vector ($\rho_{(s)}$) and rotation matrix ($R_{(s)}$) of each point along the beam w.r.t. v and u. The dependency of the terms on s are not noted in the equations throughout the text for simplicity

$$\begin{aligned} \rho_{,s} &= R(v + [0\ 0\ 1])\mathrm{d}s, \\ R_{,s} &= [u]_{\times} R, \end{aligned} \tag{1}$$

where $y_{,x} = \partial y/\partial x$ and $[x|X]_{\times} = X|x$ is an operator that creates a skew-symmetric matrix (X) from a vector x or extracts the base vector (x) of a skew-symmetric matrix X.

From a lumped-system (equivalent mass-spring-damper system) point of view, each element is a rigid cylindrical

segment with $\mathrm{d}m$ mass and $\mathrm{d}I$ inertia. The elements are interconnected with a 3-degree of freedom (DOF) linear joint, with linear compliance (translational springs) K_v, followed by a 3-DOF rotational joint with linear compliance (torsional springs) K_u. The system generalized coordinates (modeling states) for each element are $q_{(s)} = [v\ u]$. Then $\rho_{,sq}\dot{q}$ is each element total linear velocity expressed in the reference frame and $\omega_{,s}\dot{q}$ is the element total angular velocity expressed in the local frame, where $\omega_{(s)} = [R^{\mathrm{T}}R_{,q}]_{\times}$. The system continuum dynamics can be derived using the TMT method [4] as in [9],

$$\underbrace{\mathrm{d}\bar{M}\ddot{q}}_{\text{Acceleration}} + \underbrace{(\mathrm{d}D_M + \mathrm{d}D_\mu)\dot{q}}_{\text{Velocity Dependent}} + \underbrace{\mathrm{d}Kq - \mathrm{d}W}_{\text{Conservative Actions}} = \underbrace{\mathrm{d}F}_{\text{Non-conservative Actions}}, \quad (2)$$

where $\mathrm{d}\bar{M} = T^{\mathrm{T}}\mathrm{d}MT$, $T_{(s)} = [\rho_{,q}\ \omega]^{\mathrm{T}}$ is the Jacobian transformation between the Cartesian and generalized coordinates, $\mathrm{d}M = \mathrm{diag}[\mathrm{d}m\ \mathrm{d}m\ \mathrm{d}m\ \mathrm{d}I_{11}\ \mathrm{d}I_{22}\ \mathrm{d}I_{33}]$ is the system differential mass matrix, $\mathrm{d}D_M = T^{\mathrm{T}}\mathrm{d}M(T\dot{q})_{,q}$ are the inertial velocity dependent terms, $\mathrm{d}D_\mu = \mathrm{diag}[\mu_v\ \mu_u]\mathrm{d}s$ is the viscous damping term, $\mu_{v|u}$ is the element strain or curvature/torsion viscous damping coefficient, $\mathrm{d}K = \mathrm{diag}[K_v\ K_u]\mathrm{d}s$ is the element stiffness matrix based on Hooke's law, $\mathrm{d}W = \rho_{,q}^{\mathrm{T}}g\mathrm{d}m$, $g = [0\ 0\ -\mathrm{g}]$ is gravity vector, g is the gravity acceleration, $\mathrm{d}F = \mathrm{d}F_{\mathrm{p}} + \mathrm{d}F_{\mathrm{t}}$ are the non-conservative terms, e.g. due to the input pressure axial force ($f_{\mathrm{p}} = 2\Sigma_{i=1}^{3}p_i a_{\mathrm{p}}$) and bending moment ($\tau_{\mathrm{p}} = 2pa_{\mathrm{p}}\rho_{\mathrm{o}}$) as $\mathrm{d}F_{\mathrm{p}} = [0,\ 0,\ f_{\mathrm{p}},\ \tau_{\mathrm{p}}]^{\mathrm{T}}\mathrm{d}s$ in a pneumatic continuum manipulator, and external tip force (f_{t}) and moment (τ_{t}) as $\mathrm{d}F = (\rho_{,sq}^{\mathrm{T}}f_{\mathrm{t}} + u_{,q}^{\mathrm{T}}\tau_{\mathrm{t}})\mathrm{d}s$. $a_{\mathrm{p}} = \pi r_{\mathrm{p}}^2$ and r_{p} are the pneumatic chamber inner area and radius, and ρ_{o} is a matrix of which rows are position vectors of the pneumatic chambers on the manipulator cross-section as

$$\rho_{\mathrm{o}} = \begin{bmatrix} \sin(\pi/2) & -\cos(\pi/2) & 0 \\ \sin(-\pi/6) & -\cos(-\pi/6) & 0 \\ \sin(7\pi/6) & -\cos(7\pi/6) & 0 \end{bmatrix}. \quad (3)$$

The static solution to find the beam mean bent configuration, due to $f_{\mathrm{t}\epsilon}$, is found by setting $\ddot{q} = \dot{q} = f_{\mathrm{tm}} = 0$ in Eq. (2).

Eq. (1) and (2) result in a set of partial differential equations (PDE) to be integrated over time and spatial domains numerically. Optimization based methods [3], approximate continuous solutions [9] and finite element method (FEM) [11] are available for such PDEs. Alternatively, this PDE can be converted to an ordinary differential equation (ODE) by considering a fixed number of elements (n_{s}) with length $\delta s = s_{\mathrm{t}}/n_{\mathrm{s}}$ and writing Eq. (1) in a discrete form as

$$\begin{aligned} \rho_{(i+1)} &= \rho_i + R_i(v_i + [0\ 0\ 1])\delta s \\ R_{(i+1)} &= R_i + [u_i]_{\times}R_i\delta s, \end{aligned} \quad (4)$$

Thus, Eq. (2) can be derived similar to a series rigid-link mechanism with $q = [q_1\ ...\ q_{n_{\mathrm{s}}}]$ as the system states. As a result, the system static equilibrium and dynamic solution are numerically more efficient, and modal analysis and controller design tasks are easier to implement. However, the discretization is valid as long as a high enough number of elements are considered and for $K \approx \mathrm{cte}$ (constant), i.e. small axial ($\delta s \approx \mathrm{cte}$) and cross-sectional ($a \approx \mathrm{cte}$) deformations.

III. Test cases

A. Spider Web Lumped System Model

A web is modeled as a mass-spring-damper network with 8 radii and 6 circular webs to capture all the experimental points (Fig. 1- left). Point masses are assumed at the nodes and middle of the circular webs, to capture the thread second deformation modes and simulate the secondary outer frame, with Cartesian motion of the masses as generalized coordinates (q). AutoTMTDyn Matlab package is used to derive the system constrained Lagrange dynamics in a vector formalism in the form

$$M\ddot{q} + N_{\mathrm{m}}\dot{q} + L_{,q}^{\mathrm{T}}(N_{\mathrm{l}}L_{,q}\dot{q} + K\Delta L) + Mg = \lambda,\ q_{\mathrm{in}} = u, \quad (5)$$

where M is the mass matrix, N_{m} is the lateral damping matrix, N_{l} is the longitudinal damping matrix, L is the spring vector, $|L|$ is their length, $\Delta L = (1 - C_{\mathrm{c}}l_0/|L|)L_{,q}q$ is the spring deformation vector, K is the stiffness coefficient matrix, g =9.81 m/s^2 is the gravity, u is the input signal, λ is a Lagrange multiplier resulting from the input constraint, q_{in} is the generalized coordinate on which the input signal is exerted, $\dot{x} = \partial x/\partial t$ for the time derivatives, and $y_{,x} = \partial y/\partial x$ for the spatial derivatives. λ, i.e. a constrained dynamic system, is used to match the experiments excitation to numerical simulations since it is hard to measure input force on the web and only the displacement imposed at the excitation location can be observed accurately. Modeling parameters are extracted from the experimental measurements and [12]. Parts of the code used to model the web in AutoTMTDyn package and sample simulation results are presented in Fig. 1- middle & right.

B. A Bioinspired Whisker Setup

In a bioinspired whisker setup, the system is assumed as a relatively high stiffness cantilever rod as a compliant fixture to a moving base [13] (Fig. 2- left). The whisker base tissue compliance is modeled as a constant horizontal (along x-axis) linear spring (k_{b}) and a variable stiffness one (k_{s}) constrained in y- and z- axis directions. k_{b} is tuned to change the signal processing properties of the system. The whisker is rubbed against a target surface, with l_{t} offset from the whisker base, which is modeled as a fixed system contacting with a constant velocity (v_{b}) moving surface. The whisker initially deforms to a bent configuration upon contacting with the surface (Fig. 2- left), to be predicted by an initial static analysis. The surface irregularities (δl_{t}) excite small vibrations in the whisker w.r.t. its bias bent configuration. The whisker is modeled as a variable curvature (VC) continuum beam [11], with linear stress-strain relation (Hooke's law), to capture the bent configuration vibration and ability to model large deformations (structural nonlinearities) in our parameter studies. Hall Sensor (HS) readings are predicted based on the sensor varying distance to the permanent

magnet (PM) at the whisker base. The beam curvature at the contacting point with the Piezoelectric Sensor (PS) are used for modeling the PS readings. The sensors dynamics contributions are neglected in our model.

The TMT method as in section II-B is used to derive the system EOM and for static, dynamic and modal analysis. In this case, the strain ($v \approx 0$), out of plane bending ($u_{\hat{\xi}_1} \approx 0$), and torsion ($u_{\hat{\xi}_3} \approx 0$) are negligible based on the problem physics ($G \approx 0$). Parts of the code used in the "AutoTMTDyn" package input code for the intermediate elements and base stiffness variable spring are presented in Fig. 2- middle.

C. A Pneumatic Continuum Module

A Manipulator is modeled as a variable curvature (VC) continuum beam with linear stress-strain relation (Hooke's law) (Fig. 3- left). The "AutoTMTDyn" Matlab software package is used to drive the system EOM with four elements and the same parameters as in [9] except for E = 130 KPa, and linear and rotational viscous damping of 0.01 Ns/m. Parts of the input code used in The "AutoTMTDyn" package and a comparison between the simulation and experimental results in the case without external load are presented in Fig. 3- middle & right.

IV. Conclusion

In this research, new upgrades for The "AutoTMTDyn" software package, a Matlab tool for deriving the TMT Lagrange EOM in a vector formalism are introduced to facilitate the text-base user interface using a high level language, and to handle series continuum-link mechanisms. The variable curvature kinematics and the TMT Lagrange dynamics of a continuum beam are derived and the HLL elements are described and showcased in modeling of three bioinspired dynamical systems; lumped-system model of a spider web, continuum model for a rat whisker, and a continuum pneumatic module. The dynamic modeling results for the three test cases are stable and in correlation with experimental results. However, the derivation of equations and the numerical simulation are computationally expensive for such systems with large number of DOFs. The dynamic model needs to be improved by considering alternative modeling states, e.g. absolute states or momentum, to achieve sparsity for possible real-time performance. The derivation of this new system of equations is not trivial and is left for a future study. Alternatively, reduced order models [9] and modal analysis [14] can be used, instead of direct numerical simulations, to reduce the numerical computational cost of investigating the dynamical behavior of such systems.

References

[1] F. Boyer, "Multibody system dynamics for bio-inspired locomotion: from geometric structures to computational aspects," *Bioinspir. Biomim.*, p. 23, 2014.

[2] C. D. Santina, R. K. Katzschmann, A. Bicchi, and D. Rus, "Dynamic Control of Soft Robots Interacting with the Environment," (Livorno, Italy), p. 9, IEEE, 2018.

[3] S. Sadati, S. E. Naghibi, A. Shiva, I. D. Walker, K. Althoefer, and T. Nanayakkara, "Mechanics of Continuum Manipulators, a Comparative Study of Five Methods with Experiments," in *Towards Autonomous Robotic Systems*, vol. 10454, (Surrey, UK), pp. 686–702, Springer International Publishing, 2017.

[4] S. Sadati, S. Naghibi, and M. Naraghi, "An Automatic Algorithm to Derive Linear Vector Form of Lagrangian Equation of Motion with Collision and Constraint," *Procedia Computer Science*, vol. 76, pp. 217–222, 2015.

[5] M. Wisse and R. Q. v. d. Linde, *Delft Pneumatic Bipeds*. Springer Science & Business Media, June 2007. Google-Books-ID: NE8CmUMdG38C.

[6] "Robotics System Toolbox," 2018.

[7] D. Negrut and A. Dyer, "Adams/solver primer," *MSC. Software Documentation, Ann Arbor*, 2004.

[8] S. Sadati and A. Meghdari, "Singularity-free planning for a robot cat free-fall with control delay: Role of limbs and tail," in *2017 8th International Conference on Mechanical and Aerospace Engineering (ICMAE)*, pp. 215–221, July 2017.

[9] S. Sadati, S. E. Naghibi, I. D. Walker, K. Althoefer, and T. Nanayakkara, "Control Space Reduction and Real-Time Accurate Modeling of Continuum Manipulators Using Ritz and RitzGalerkin Methods," *IEEE Robotics and Automation Letters*, vol. 3, pp. 328–335, Jan. 2018.

[10] S. Sadati, "AutoTMTDyn Software Package," May 2017. https://github.com/hadisdt/AutoTMTDyn.

[11] M. Gazzola, L. H. Dudte, A. G. McCormick, and L. Mahadevan, "Forward and inverse problems in the mechanics of soft filaments," *Royal Society Open Science*, vol. 5, p. 171628, June 2018.

[12] S. Sadati, S. E. Naghibi, K. Althoefer, and T. Nanayakkara, "Toward a Low Hysteresis Helical Scale Jamming Interface Inspired by Teleost Fish Scale Morphology and Arrangement," (Livorno, Italy), p. 7, IEEE, 2018.

[13] H. Wegiriya, N. Sornkarn, H. Bedford, and T. Nanayakkara, "A biologically inspired multimodal whisker follicle," pp. 003847–003852, IEEE, Oct. 2016.

[14] C. Della Santina, D. Lakatos, A. Bicchi, and A. Albu-Schffer, "Using Nonlinear Normal Modes for Execution of Efficient Cyclic Motions in Soft Robots," *arXiv:1806.08389 [cs]*, June 2018. arXiv: 1806.08389.