2008 IEEE/RSJ International Conference on Intelligent Robots and Systems
Acropolis Convention Center
Nice, France, Sept, 22-26, 2008

A Modular Software Architecture for Simulating Mechanical Systems Involving Coulomb Friction Integrable by the Runge-Kutta Method

Ryo Kikuuwe and Motoji Yamamoto

Abstract— **This paper presents a modular architecture of simulation software that is capable of properly capturing Coulomb friction in mechanical systems. The presented architecture is built upon Kikuuwe et al.'s discrete-time friction model, which is based on the implicit Euler method. The paper reformulates the model into a form that can be integrable through the standard fourth order Runge-Kutta method. Some examples based on the presented architecture are presented.**

I. Introduction

Almost all mechanical systems contain frictional contacts among objects. Most simple representation of friction is the Coulomb friction model, which is described as follows:

$$f = Fv/|v| \tag{1}$$

where f is the friction force, v is the relative velocity, and $F > 0$ is the magnitude of the kinetic friction force. In this model, the force is discontinuous with respect to the velocity v, and the force balances to external forces at zero velocity so as to maintain zero velocity. A classical approach to treat this discontinuity is to use a non-physical, artificial threshold below which the velocity is considered zero [1]. As is well known, this approach produces either of chattering due to repeated zero-velocity crossing or unbounded drift below the threshold velocity. Some modern friction models [2]–[4] employ differential equations to describe continuous transitions from static friction to kinetic friction. These models are difficult to be used in multidimensional space or exhibit unbounded drift even when the applied force is below the static friction level.

Recently, Kikuuwe et al. [5] proposed a discrete-time friction model that is free from the undesirable properties mentioned above. The idea behind the model is that the troublesome discontinuous dynamics described as (1) is coupled with a viscoelastic dynamics through a massless object, as illustrated in Fig. 1. Such a physical representation is indeed common among some of modern friction models, but Kikuuwe et al.'s model is different in its mathematical formulation; in their model, the system of Fig. 1 is described as the following set of differential algebraic constraints:

$$f = F(v - \dot{e})/|v - \dot{e}| \tag{2a}$$
$$f = Ke + B\dot{e}. \tag{2b}$$

Here, $K > 0$ and $B > 0$ are stiffness and viscosity, respectively, of the viscoelastic dynamics, and e is the displacement of the viscoelastic element. In [5], the analytical solution for the constraint (2) is derived based on the implicit (backward) Euler scheme, and it is formulated as an algorithm that accepts velocity input v and produces force output f. The model does not exhibit drift, and does not exhibit chattering as long as K and B are appropriately chosen. Moreover, rate-dependent friction laws such as viscous and Stribeck effects can also be included, as detailed in [5]. A more detailed literature review supporting this friction model can be found in [5]. The model is straightforward to be used in forward-Euler simulations. Thus, the model is useful for physics-based animation and haptic rendering, in which the accuracy is less counted than visual/haptic realism.

This paper aims to broaden the application of the previously-proposed Kikuuwe et al.'s friction model toward the field of general physical simulation in which the accuracy is of more importance. In such applications, the fourth order Runge-Kutta method (RK4) is much more preferred than the Euler methods. A technical problem arising here is that RK4 requires the system to be described in a differential equation in continuous time, but Kikuuwe et al.'s model is described in the form of difference (recurrence) equation in discrete time. Moreover, RK4 requires "trial" computations of the derivatives, which do not update the state vector. Thus, some modifications are required for Kikuuwe et al.'s friction model to allow its use in the RK4 framework.

Another problem in using Kikuuwe et al.'s model with RK4 is that the model is a *modular* model although RK4 can be viewed as a *holistic* method. Kikuuwe et al.'s friction model is originally formulated as an *impedance*-type (motion-in, force-out) software component, which is meant to be used in a modular software architecture composed of impedance and *admittance* (force-in, motion-out) components. The use of RK4 in a modular software architecture is not straightforward because RK4 usually requires the whole system to be described as a single differential equation.

The rest of this paper is organized as follows. Section II prepares a modular architecture of a mechanical system simulator that can be integrated through RK4, which can be

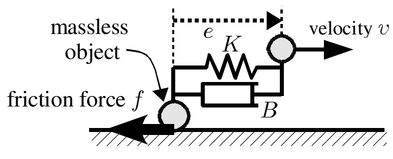

Fig. 1. The physical model upon which Kikuuwe et al.'s friction model [5] is built. The novelty of their friction model lies in the mathematical formulation of this physical model, not in this physical model itself.

The authors are with Kyushu University, Fukuoka 819-0395, Japan. E-mail: kikuuwe@ieee.org

built up from admittance and impedance components. The main contribution of the paper is described in section III, which presents a modification of Kikuuwe et al.'s friction model as an impedance component integrable through RK4, preserving its original advantages. Section IV presents example applications of the presented technique. Section V provides the concluding remarks.

II. Modular Representation of Dynamic Systems Integrable By RK4

A. Fourth Order Runge-Kutta Method

Consider a dynamic system described as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{\mathcal{F}}(\boldsymbol{x}, t) \tag{3}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the system's state vector, $t \in \mathbb{R}$ is the time, $\boldsymbol{\mathcal{F}}$ is a function from $\mathbb{R}^n \times \mathbb{R}$ to $\mathbb{R}^n$, and $\dot{\boldsymbol{x}} = d\boldsymbol{x}/dt \in \mathbb{R}^n$. Let $\boldsymbol{x}(k)$ be the value of $\boldsymbol{x}$ at time $t = kT$ where k is an integer index and T is the timestep size. The most straightforward method to obtain $\boldsymbol{x}(k+1)$ is the forward Euler approach, which is described as follows:

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + T\boldsymbol{\mathcal{F}}(\boldsymbol{x}(k), kT). \tag{4}$$

This scheme usually provides inaccurate results and thus is not used for simulation purposes except in cases where the speed of computation is of the primary importance.

The RK4, which is described below, provides a more accurate result:

For $r = 1, 2, 3, 4$ (5a)

$$\boldsymbol{\xi}_r = \boldsymbol{\mathcal{F}}(\boldsymbol{x}(k) + T_{r-1}\boldsymbol{\xi}_{r-1}, kT + T_{r-1}) \tag{5b}$$

End For (5c)

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + (T/6)(\boldsymbol{\xi}_1 + 2\boldsymbol{\xi}_2 + 2\boldsymbol{\xi}_3 + \boldsymbol{\xi}_4) \tag{5d}$$

where $\{T_0, T_1, T_2, T_3\} = \{0, T/2, T/2, T\}$. For brevity of notation, we define $\boldsymbol{\xi}_0 = \boldsymbol{o}$. This technique requires four computations of the function $\boldsymbol{\mathcal{F}}$ to increment the time for one timestep size. Here, we should be aware that all state variables in the system must be included in the vector $\boldsymbol{x}$ and that an analytical expression of function $\boldsymbol{\mathcal{F}}$ must be derived before the software is written.

B. Modular Architecture

For extensible, reusable software architecture, a mechanical system should be described as a network of various software components, each of which has a particular physical counterpart in the simulated mechanical system. In such software architecture, it is convenient to use two types of components: the admittance and impedance types. An admittance component is a component that accepts force as the input and produces position and velocity as the output. An impedance component is a component that accepts position input and produces force output. Yoshikawa and Ueda [6] have also presented a similar architecture, in which admittance and impedance components are termed as 'dynamics modules' and 'interaction modules,' respectively.

In software of a modular architecture, a component updates its own state variables according to some inputs. In the RK4 scheme described in (5), however, all the state variables of the whole system are included in a single vector $\boldsymbol{x}$, and the analytical expression of the whole system should be given in a single differential equation. Thus, a care should be taken for the basic architecture of the software to combine a modular architecture and the RK4 integration.

A possible solution follows. A general expression of an admittance component can be described as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{\mathcal{A}}(\boldsymbol{x}, \hat{\boldsymbol{u}}). \tag{6}$$

Here, $\boldsymbol{x}$ is the state vector of the component, which consists of the information of generalized position/velocity, and $\hat{\boldsymbol{u}}$ is the input to the component, which consists of the information of generalized force. Eq. (6) implies that how $\boldsymbol{x}$, which represents the position and velocity, changes is determined by the force input $\hat{\boldsymbol{u}}$. One simple example is a point mass $M > 0$ accepting a translational force $\boldsymbol{f} \in \mathbb{R}^3$, which can be described as follows:

$$\begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{v}} \end{bmatrix} = \boldsymbol{\mathcal{A}}_{\text{mass}}\left(\begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{v} \end{bmatrix}, \boldsymbol{f}\right) = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{f}/M \end{bmatrix} \tag{7}$$

where $\boldsymbol{p} \in \mathbb{R}^3$ and $\boldsymbol{v} \in \mathbb{R}^3$ are the position and velocity, respectively, of the mass point. Another example is an n-DOF robotic system, of which the equation of motion is given in the form of $\boldsymbol{\tau} = \boldsymbol{H}(\boldsymbol{\theta})\dot{\boldsymbol{\omega}} + \boldsymbol{c}(\boldsymbol{\theta}, \boldsymbol{\omega})$ where $\boldsymbol{\theta} \in \mathbb{R}^n$ is the vector consisting of the joint angles, $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector consisting of the joint torques, $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ is the inertia matrix, $\boldsymbol{c} \in \mathbb{R}^n$ is Coriolis and centrifugal forces, and $\boldsymbol{\omega} = \dot{\boldsymbol{\theta}}$. This robotic system can be described as an admittance component as follows:

$$\begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \boldsymbol{\mathcal{A}}_{\text{rob}}\left(\begin{bmatrix} \boldsymbol{\theta} \\ \boldsymbol{\omega} \end{bmatrix}, \boldsymbol{\tau}\right) = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{H}(\boldsymbol{\theta})^{-1}(\boldsymbol{\tau} - \boldsymbol{c}(\boldsymbol{\theta}, \boldsymbol{\omega})) \end{bmatrix} \tag{8}$$

Recent techniques on efficient computation of forward dynamics of robotic systems, e.g., [7], can be embedded in this framework as a function $\boldsymbol{\mathcal{A}}_{\text{rob}}$.

An impedance component, on the other hand, can be described in the following general form:

$$\begin{bmatrix} \boldsymbol{u} \\ \dot{\boldsymbol{y}} \end{bmatrix} = \boldsymbol{\mathcal{I}}(\hat{\boldsymbol{x}}, \boldsymbol{y}) \tag{9}$$

where $\hat{\boldsymbol{x}}$ is the generalized position/velocity input from external sources, $\boldsymbol{u}$ is the generalized force produced by the component. The component may or may not hold some state variables, which are included in $\boldsymbol{y}$ if any. A spring-damper element can be described as an impedance component as follows:

$$\boldsymbol{f} = \boldsymbol{\mathcal{I}}_{\text{sd}}\left(\begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{v} \end{bmatrix}\right) = K\boldsymbol{p} + B\boldsymbol{v} \tag{10}$$

where $\boldsymbol{p}$ and $\boldsymbol{v}$ are displacement and relative velocity, respectively, $K > 0$ and $B > 0$ are the stiffness and viscosity coefficients, respectively, and $\boldsymbol{f}$ is the force. Some modern models of friction are also described as this class of components. Dahl model [2] describes the friction as follows

$$\begin{bmatrix} f \\ \dot{e} \end{bmatrix} = \boldsymbol{\mathcal{I}}_{\text{fricD}}(v, e) = \begin{bmatrix} Ke \\ v - |v|Ke/F \end{bmatrix} \tag{11}$$

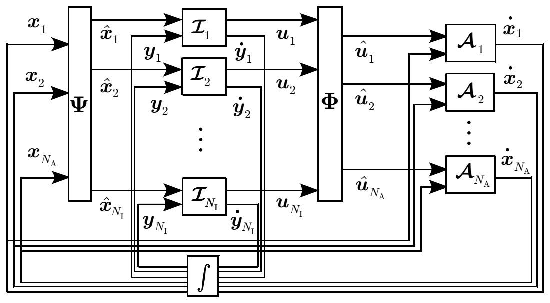

Fig. 2. A mechanical system composed of multiple impedance and admittance components.

where v is the relative velocity, f is the friction force, $F > 0$ is the magnitude of the kinetic friction force, K is the initial elasticity in the static friction state, and e is the state variable that can be interpreted as the elastic displacement. LuGre model [3], which is a generalization of Dahl model, is describes as follows:

$$\begin{bmatrix} f \\ \dot{e} \end{bmatrix} = \boldsymbol{\mathcal{I}}_{\text{fricL}}(v, e) = \begin{bmatrix} Ke + B(v - |v|e/g(v)) + Dv \\ v - |v|e/g(v) \end{bmatrix} \tag{12}$$

where $B > 0$, $D > 0$, and $g(v) > 0$ is a function that allows the dependence of the friction force on the relative velocity (for more detail, see [3]). As is discussed in detail in [5], these friction models suffer from unbounded drift or difficulty in multidimensional extensions.

Let us consider a system consisting of N_I impedance components and N_A admittance components. The output (generalized force) produced by the impedance components are provided to the admittance components. The state vectors (generalized position/velocity) of the admittance and impedance components are integrated through RK4. Based on the basic procedure (5) of RK4, the state vectors of the components can be updated through the following procedure:

For $r = 1, \cdots, 4$ (13a)

  For $i = 1, \cdots, N_I$; $\boldsymbol{y}_i = \boldsymbol{y}_i(k) + T_{r-1}\boldsymbol{\eta}_{i,r-1}$ (13b)

  For $j = 1, \cdots, N_A$; $\boldsymbol{x}_j = \boldsymbol{x}_j(k) + T_{r-1}\boldsymbol{\xi}_{j,r-1}$ (13c)

$$\begin{bmatrix} \hat{\boldsymbol{x}}_1 \\ \vdots \\ \hat{\boldsymbol{x}}_{N_I} \end{bmatrix} = \boldsymbol{\Psi}\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{N_A} \end{bmatrix}\right) \tag{13d}$$

  For $i = 1, \cdots, N_I$; $\begin{bmatrix} \boldsymbol{u}_i \\ \boldsymbol{\eta}_{i,r} \end{bmatrix} = \boldsymbol{\mathcal{I}}_i(\hat{\boldsymbol{x}}_i, \boldsymbol{y}_i)$ (13e)

$$\begin{bmatrix} \hat{\boldsymbol{u}}_1 \\ \vdots \\ \hat{\boldsymbol{u}}_{N_A} \end{bmatrix} = \boldsymbol{\Phi}\left(\begin{bmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{N_I} \end{bmatrix}\right) \tag{13f}$$

  For $j = 1, \cdots, N_A$; $\boldsymbol{\xi}_{j,r} = \boldsymbol{\mathcal{A}}_j(\boldsymbol{x}_j, \hat{\boldsymbol{u}}_j)$ (13g)

End For (13h)

For $i = 1, \cdots, N_I$; (13i)

$$\boldsymbol{y}_i(k+1) = \boldsymbol{y}_i(k) + \frac{T}{6}(\boldsymbol{\eta}_{i,1} + 2\boldsymbol{\eta}_{i,2} + 2\boldsymbol{\eta}_{i,3} + \boldsymbol{\eta}_{i,4}) \tag{13j}$$

For $j = 1, \cdots, N_A$; (13k)

$$\boldsymbol{x}_j(k+1) = \boldsymbol{x}_j(k) + \frac{T}{6}(\boldsymbol{\xi}_{j,1} + 2\boldsymbol{\xi}_{j,2} + 2\boldsymbol{\xi}_{j,3} + \boldsymbol{\xi}_{j,4}) \tag{13l}$$

Here, the functions $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ represent some computations such as coordinate transformations. For example, the input to a spring-damper element $\boldsymbol{\mathcal{I}}_{\text{sd}}$, which is displacement, can be computed by taking the difference between the outputs, which are positions, of two masses $\boldsymbol{\mathcal{A}}_{\text{mass}}$.

Fig. 2 schematically illustrates the architecture (13). This structure is suited for parallel computation as the computations within each "For" loop in (13) are order insensitive.

III. Incorporating Friction

A. Kikuuwe et al.'s Friction Model

Now we are in position to discuss the problem of implementing Kikuuwe et al.'s friction model [5] in the RK4-integrable modular architecture (13). The friction model has the following structure:[1]

$$\boldsymbol{f}(k) = \mathbf{gsat}(F, (B + TK)\boldsymbol{v}(k) + K\boldsymbol{e}(k-1)) \tag{14a}$$
$$\boldsymbol{e}(k) = (B\boldsymbol{e}(k-1) + T\boldsymbol{f}(k))/(B + TK) \tag{14b}$$

Here, $K > 0$, $B > 0$, $F > 0$, $\boldsymbol{v}$ is the input velocity, and $\boldsymbol{f}$ is the output force. The variable $\boldsymbol{e}$ is the state variable. The function $\mathbf{gsat} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\mathbf{gsat}(X, \boldsymbol{x}) = \begin{cases} \boldsymbol{x} & \text{if } \|\boldsymbol{x}\| \le X \\ X\boldsymbol{x}/\|\boldsymbol{x}\| & \text{if } \|\boldsymbol{x}\| > X, \end{cases} \tag{15}$$

which imposes a saturation at the level of X to the argument variable. The expression (14) is algebraically equivalent to the following equation[2]:

$$\boldsymbol{f}(k) = F\mathbf{sgn}(\boldsymbol{v}(k) - (\boldsymbol{e}(k) - \boldsymbol{e}(k-1))/T) \tag{16a}$$
$$\boldsymbol{f}(k) = K\boldsymbol{e}(k) + B(\boldsymbol{e}(k) - \boldsymbol{e}(k-1))/T, \tag{16b}$$

which is the backward-Euler discretization of

$$\boldsymbol{f} = F\mathbf{sgn}(\boldsymbol{v} - \dot{\boldsymbol{e}}), \ \boldsymbol{f} = K\boldsymbol{e} + B\dot{\boldsymbol{e}}. \tag{17}$$

Here, the function $\mathbf{sgn}$ is the multidimensional version of the signum function, which is defined as

$$\mathbf{sgn}(\boldsymbol{x}) \begin{cases} = \boldsymbol{x}/\|\boldsymbol{x}\| & \text{if } \|\boldsymbol{x}\| \ne 0 \\ \in \{\boldsymbol{e} \in \mathbb{R}^n \mid \|\boldsymbol{e}\| \le 1\} & \text{if } \|\boldsymbol{x}\| = 0. \end{cases} \tag{18}$$

Thus, (14) can be physically interpreted as Fig. 1.

For simulating friction between two rigid bodies, the stiffness coefficient K should be chosen as high as permitted by the stability of the simulated system, which is influenced by the timestep size T. The viscosity coefficient B also should be chosen high to reduce the oscillation. For simulating friction between soft objects, the parameters K and B can be chosen according to the lateral stiffness and viscosity of the simulated objects.

[1]For the simplicity of discussion, this paper ignores rate-dependent friction phenomena such as viscous and Stribeck effects. It is however possible to include such phenomena as detailed in section III.F of [5].

[2]The equivalence between (14) and (16) can be proven by using the relation $\boldsymbol{y} = X\mathbf{sgn}(\boldsymbol{x} - \boldsymbol{y}) \iff \boldsymbol{y} = \mathbf{gsat}(X, \boldsymbol{x})$, $X > 0$, of which the details is explained in some of previous papers [8].

B. Modification for RK4 Integration

In order to use the friction model (14) in the framework of (13), the model should be rewritten in the form of the function $\boldsymbol{\mathcal{I}}$ in (9). Because the right-hand side of (9) contains the derivative of the state variable, $\dot{\boldsymbol{y}}$, (14) should be modified to compute some values associated with the derivative of $\boldsymbol{e}$. One solution is to use the function of the following definition:

```
Function $\boldsymbol{\mathcal{I}}_{\text{fricK}}(\boldsymbol{v}, \boldsymbol{e};\, \tau)$      (19a)
    $\boldsymbol{f} = \mathbf{gsat}(F, (B + \tau K)\boldsymbol{v} + K\boldsymbol{e})$      (19b)
    $\boldsymbol{\varepsilon} = (\boldsymbol{f} - K\boldsymbol{e})/(B + \tau K)$      (19c)
    Return $[\boldsymbol{f}^T,\ \boldsymbol{\varepsilon}^T]^T$      (19d)
End Function      (19e)
```

By using this function, (14) can be rewritten as follows:

$$\begin{bmatrix} \boldsymbol{f}(k) \\ \boldsymbol{\varepsilon}(k) \end{bmatrix} = \boldsymbol{\mathcal{I}}_{\text{fricK}}(\boldsymbol{v}(k), \boldsymbol{e}(k-1);\, T) \tag{20a}$$

$$\boldsymbol{e}(k) = \boldsymbol{e}(k-1) + T\boldsymbol{\varepsilon}(k). \tag{20b}$$

It is important to be aware that the function $\boldsymbol{\mathcal{I}}_{\text{fricK}}$ takes the timestep size τ as an argument. This fact leads to a need to slightly generalize RK4 (5) as follows:

```
For $r = 1, 2, 3, 4$      (21a)
    $\boldsymbol{\xi}_r = \boldsymbol{\mathcal{F}}(\boldsymbol{x}(k) + T_{r-1}\boldsymbol{\xi}_{r-1}, kT + T_{r-1}; T_r)$      (21b)
End For      (21c)
$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + (T/6)(\boldsymbol{\xi}_1 + 2\boldsymbol{\xi}_2 + 2\boldsymbol{\xi}_3 + \boldsymbol{\xi}_4)$      (21d)
```

where $\{T_0, T_1, T_2, T_3, T_4\} = \{0, T/2, T/2, T, T\}$. Accordingly, in the algorithm (13), (13e) should be replaced by

$$\text{For } i = 1, \cdots, N_I;\ \begin{bmatrix} \boldsymbol{u}_i \\ \boldsymbol{\eta}_{i,r} \end{bmatrix} = \boldsymbol{\mathcal{I}}_i(\hat{\boldsymbol{x}}_i, \boldsymbol{y}_i;\, T_r)\,. \tag{22}$$

The function $\boldsymbol{\mathcal{I}}_{\text{fricK}}$ of (19) guarantees $\boldsymbol{\varepsilon} = \boldsymbol{v}$ as long as $\|(B + \tau K)\boldsymbol{v} + K\boldsymbol{e}\| < F$. This implies that, as long as $\|(B + \tau K)\boldsymbol{v} + K\boldsymbol{e}\| < F$ is satisfied in the four iterations of the RK4, the "massless object" in Fig. 1 stays at a fixed position because $\int(\boldsymbol{\varepsilon} - \boldsymbol{v})dt = \boldsymbol{p} - \boldsymbol{e}$ remains constant. Due to this characteristic, this algorithm is able to capture static friction without exhibiting drift.

The function $\boldsymbol{\mathcal{I}}_{\text{fricK}}$ does not include the influence of the normal force. Based on (14), Coulomb friction on a flat surface can be easily derived as follows:

$$f_z(k) = \begin{cases} Kp_z(k) + Bv_z(k) & \text{if } p_z(k) < 0 \\ 0 & \text{if } p_z(k) \geq 0 \end{cases} \tag{23a}$$

$$\boldsymbol{f}^*_{xy}(k) = (B + TK)\begin{bmatrix} v_x(k) \\ v_y(k) \end{bmatrix} + K\boldsymbol{e}(k-1) \tag{23b}$$

$$\boldsymbol{f}_{xy}(k) = \begin{bmatrix} f_x(k) \\ f_y(k) \end{bmatrix} = \mathbf{gsat}(-\mu f_z(k), \boldsymbol{f}^*_{xy}(k)) \tag{23c}$$

$$\boldsymbol{e}(k) = (B\boldsymbol{e}(k-1) + T\boldsymbol{f}_{xy}(k))/(B + TK) \tag{23d}$$

where $\mu > 0$ is the friction coefficient, $\boldsymbol{f} = [f_x, f_y, f_z]^T$, $\boldsymbol{v} = [v_x, v_y, v_z]^T$, and $\boldsymbol{e} \in \mathbb{R}^2$. The flat surface is assumed to include the origin and to be normal to z axis. The algorithm (23) can be easily rewritten in the following form:

$$\begin{bmatrix} \boldsymbol{f}(k) \\ \boldsymbol{\varepsilon}(k) \end{bmatrix} = \boldsymbol{\mathcal{I}}_{\text{fricKN}}\left(\begin{bmatrix} p_z(k) \\ \boldsymbol{v}(k) \end{bmatrix}, \boldsymbol{e}(k-1);\, T\right) \tag{24a}$$

$$\boldsymbol{e}(k) = \boldsymbol{e}(k-1) + T\boldsymbol{\varepsilon}(k) \tag{24b}$$

with the function $\boldsymbol{\mathcal{I}}_{\text{fricKN}} : \mathbb{R}^4 \times \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^5$ of an appropriate definition.

IV. Examples

A. Simulation I: A Rolling Sphere With Slip

As an example for dynamics systems involving friction, we consider a rigid spherical object rolling or slipping on a fixed flat surface, as illustrated in Fig. 4. This system can be described as a combination of one admittance component and one impedance component. The admittance component represents the dynamics of the spherical object, while the impedance component represents the frictional contact between the object and the fixed surface.

The state vector of the admittance component is

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}^T & \boldsymbol{v}^T & \boldsymbol{q}^T & \boldsymbol{\omega}^T \end{bmatrix}^T \in \mathbb{R}^{13}. \tag{25}$$

Here, $\boldsymbol{p} \in \mathbb{R}^3$ and $\boldsymbol{v} \in \mathbb{R}^3$ are the position and the velocity, respectively, of the gravity center of the object. The vector $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity and $\boldsymbol{q}$ is the unit-quaternion representation (detailed in, e.g., [9]) of the attitude of the object. The function $\boldsymbol{\mathcal{A}}$ for this object can be obtained based on the basic Newton and Euler's equations of motion, which is written as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{\mathcal{A}}_{\text{rigid}}\left(\boldsymbol{x}, \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix}\right) = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{f}/M \\ \boldsymbol{Q}(\boldsymbol{\omega}, \boldsymbol{q}) \\ \boldsymbol{J}^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega}) \end{bmatrix} \tag{26}$$

Here, $\boldsymbol{f} \in \mathbb{R}^3$ and $\boldsymbol{\tau} \in \mathbb{R}^3$ are the translational force and the moment applied to the object, and $M = 0.3$ kg and $\boldsymbol{J} \in \mathbb{R}^{3\times3}$ are the mass value and the inertia matrix, respectively, of the object. The function $\boldsymbol{Q} : \mathbb{R}^3 \times \mathbb{R}^4 \to \mathbb{R}^4$ denotes the function that transforms the angular velocity into the quaternion rate; $\dot{\boldsymbol{q}} = \boldsymbol{Q}(\boldsymbol{\omega}, \boldsymbol{q})$. Let $\boldsymbol{f}_F$ be the force acting from the flat surface to the contact point. Then, $\boldsymbol{f}$ and $\boldsymbol{\tau}$ in (26) satisfy $\boldsymbol{f} = \boldsymbol{f}_F + M\boldsymbol{g}$ and $\boldsymbol{\tau} = \boldsymbol{a}_z \times \boldsymbol{f}_F$ where $\boldsymbol{a}_z = [0, 0, -R]^T$, $R = 0.2$ m is the radius of the object, and $\boldsymbol{g} = [0, 0, 9.8]^T$ m/s^2 is the gravitational acceleration.

The force $\boldsymbol{f}_F$ can be obtained through the impedance component $\boldsymbol{\mathcal{I}}_{\text{fricKN}}$, which is used in (24) and defined through (23). The input into $\boldsymbol{\mathcal{I}}_{\text{fricKN}}$ is the velocity of the contact point, which is given by $\boldsymbol{v}_c = \boldsymbol{v} + \boldsymbol{\omega} \times \boldsymbol{a}_z$. The parameters for $\boldsymbol{\mathcal{I}}_{\text{fricKN}}$ were set to be $K = 10^7$ N/m, $B = 10$ Ns/m, and $\mu = 0.1$.

Then, the algorithm in one timestep is written as follows:

```
For $r = 1, 2, 3, 4$      (27a)
    $\boldsymbol{e} = \boldsymbol{e}(k) + \boldsymbol{\varepsilon}_{r-1}T_{r-1}$      (27b)
    $\boldsymbol{x} = \boldsymbol{x}(k) + \boldsymbol{\xi}_{r-1}T_{r-1}$      (27c)
    $\begin{bmatrix} \boldsymbol{f}_F \\ \boldsymbol{\varepsilon}_r \end{bmatrix} = \boldsymbol{\mathcal{I}}_{\text{fricKN}}\left(\begin{bmatrix} p_z - R \\ \boldsymbol{v} + \boldsymbol{\omega} \times \boldsymbol{a}_z \end{bmatrix}, \boldsymbol{e};\, T_r\right)$      (27d)
    $\boldsymbol{\xi}_r = \boldsymbol{\mathcal{A}}_{\text{rigid}}\left(\boldsymbol{x}, \begin{bmatrix} \boldsymbol{f}_F + M\boldsymbol{g} \\ \boldsymbol{a}_z \times \boldsymbol{f}_F \end{bmatrix}\right)$      (27e)
End For      (27f)
```

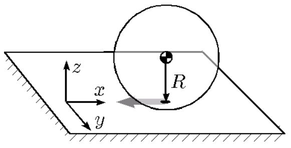

Fig. 3. Simulation I: A spherical object on a frictional flat surface.

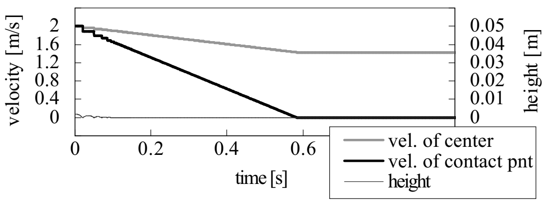

Fig. 4. Results of simulation I.

$$e(k+1) = e(k) + (T/6)(\varepsilon_1 + 2\varepsilon_2 + 2\varepsilon_3 + \varepsilon_4) \quad (27g)$$

$$x(k+1) = x(k) + (T/6)(\xi_1 + 2\xi_2 + 2\xi_3 + \xi_4) \quad (27h)$$

The timestep size was set to be $T = 10^{-4}$ s.

At $t = 0$, the state of the object were set to be $\boldsymbol{v} = [2,0,0]^T$ m/s and $\boldsymbol{p} = [0,0,1.01R]^T$. The result is shown in Fig. 4, in which the horizontal velocity of the gravity center, the horizontal velocity of the contact point, and the height of the object. The velocity of the contact point reaches zero at $t = 0.58$ s, which implies that the sphere reaches the pure rolling state without slipping. These results indicate the validity of the representation $\mathcal{I}_{\text{fricKN}}$ in RK4 framework.

B. Simulation II: Rigid Link Mechanism: Constraint-Based Formulation

As a second example, we consider a two-link serial mechanism illustrated in Fig. 5. Both of the two joints are assumed to be subject to Coulomb friction and be actuated. This system can be described as a combination of one admittance component and two impedance components. The admittance component represents the dynamics of the link mechanism, and each of the impedance components represents the Coulomb friction in each joint.

The admittance component representing the link mechanism can be formulated as $\mathcal{A}_{\text{rob}}$ in (8), in which the vectors $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ are defined as $\boldsymbol{\theta} = [\theta_0, \theta_1]^T$ and $\boldsymbol{\omega} = [\omega_0, \omega_1]^T$ where θ_i and ω_i are the angle and the angular velocity, respectively, of the i-th joint. The input to $\mathcal{A}_{\text{rob}}$ is $\boldsymbol{\tau} = [\tau_0, \tau_1]^T$ where τ_i is the torque applied to the i-th joint, which are sum of the friction torque and the actuator torque. The friction torque for the i-th joint, $\tau_{F,i}$, can be obtained by the function $\mathcal{I}_{\text{fricK}}$ in (19) with the input being ω_i, where $K = 5000$ Nm/rad, $B = 1$ Nms/rad, and $F = 0.5$ Nm.

In the simulation, the i-th joint ($i \in \{0.1\}$) was actuated by the time-varying torque $\tau_{M,i}$ that is illustrated in Fig. 6. The purpose of this is to test the drift characteristics of friction models, as demonstrated in [4]. The input to $\mathcal{A}_{\text{rob}}$ was determined as $\tau_i = \tau_{F,i} + \tau_{M,i}$. The algorithm in one timestep is described as follows:

$$\text{For } r = 1, 2, 3, 4 \quad (28a)$$

$$t = kT + T_{r-1} \quad (28b)$$

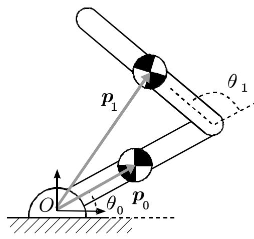

Fig. 5. A serial link mechanism used in simulations II and III.

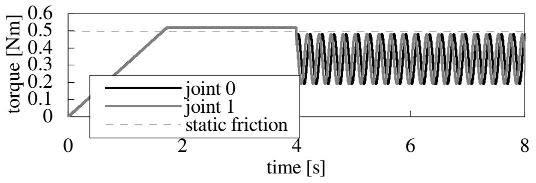

Fig. 6. Applied torque in simulation II and III.

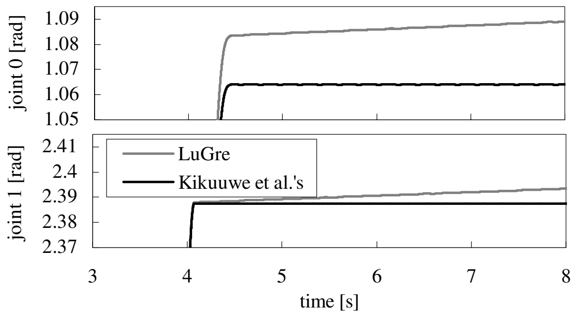

Fig. 7. Results of simulation II.

$$\tau_{M,0} = \begin{cases} \min(0.52, 0.3t) & \text{if } t < 4 \\ 0.336 + 0.144\sin(100t) & \text{otherwise} \end{cases} \quad (28c)$$

$$\tau_{M,1} = \begin{cases} \min(0.52, 0.3t) & \text{if } t < 4 \\ 0.336 + 0.144\cos(100t) & \text{otherwise} \end{cases} \quad (28d)$$

$$e_0 = e_0(k) + \varepsilon_{0,r-1}T_{r-1} \quad (28e)$$

$$e_1 = e_1(k) + \varepsilon_{1,r-1}T_{r-1} \quad (28f)$$

$$x = x(k) + \xi_{0,r-1}T_{r-1} \quad (28g)$$

$$\begin{bmatrix} \tau_{F,0} \\ \varepsilon_{0,r} \end{bmatrix} = \mathcal{I}_{\text{fricK}}(\omega_0, e_0;\, T_r) \quad (28h)$$

$$\begin{bmatrix} \tau_{F,1} \\ \varepsilon_{1,r} \end{bmatrix} = \mathcal{I}_{\text{fricK}}(\omega_1, e_1;\, T_r) \quad (28i)$$

$$\xi_i = \mathcal{A}_{\text{rob}}\left(x, \begin{bmatrix} \tau_{M,0} + \tau_{F,0} \\ \tau_{M,1} + \tau_{F,1} \end{bmatrix}\right) \quad (28j)$$

$$\text{End For} \quad (28k)$$

$$e_0(k+1) = e_0(k) + (T/6)(\varepsilon_{0,1} + 2\varepsilon_{0,2} + 2\varepsilon_{0,3} + \varepsilon_{0,4}) \quad (28l)$$

$$e_1(k+1) = e_1(k) + (T/6)(\varepsilon_{1,1} + 2\varepsilon_{1,2} + 2\varepsilon_{1,3} + \varepsilon_{1,4}) \quad (28m)$$

$$x(k+1) = x(k) + (T/6)(\xi_1 + 2\xi_2 + 2\xi_3 + \xi_4) \quad (28n)$$

where $\boldsymbol{x} = [\boldsymbol{\theta}^T, \boldsymbol{\omega}^T]^T = [\theta_0, \theta_1, \omega_0, \omega_1]^T$. The timestep size was set to be $T = 10^{-4}$ s.

Fig. 7 shows the results. The mechanism stops at around $t = 4.5$ s as the actuator torques become smaller than the static friction level. For comparison, we also performed another set of simulation using LuGre model, replacing

$\mathcal{I}_{\text{fricK}}$ in (28) by $\mathcal{I}_{\text{fricL}}$ in (12). With LuGre model, the link angles drift; this is a well-known characteristic of LuGre model. It has been shown in the previous paper [5] that such spurious behavior does not appear with Kikuuwe et al.'s model in Euler integration. The present results show that this property of Kikuuwe et al.'s friction model is preserved even after the modification described in section III.

C. Simulation III: Rigid Link Mechanism: Penalty-Based Formulation

The link mechanism of Fig. 5 can be modeled also in a penalty-based approach. In this approach, each of the links is modeled as a rigid body, as $\mathcal{A}_{\text{rigid}}$ in (26), and each of the joints is modeled as a stiff spring-damper element that produces forces and moments to *penalize* the separation between the links and the bending of the joint axes.

Let $\boldsymbol{x}_i \in \mathbb{R}^{13}$ $(i = 0, 1)$ be the state of the link i in the form of (25). Let $\boldsymbol{x}_{iB}$ $(i = 0, 1)$ be the state (in the format of (25)) of the base-side articulation point of the link i, and $\boldsymbol{x}_{iE}$ be that of the tip-side articulation point of the link i. The vector $\boldsymbol{x}_{iB}$ and $\boldsymbol{x}_{iE}$ are obtained through a simple coordinate transformation of $\boldsymbol{x}_i$. The joint i can be modeled as an impedance component, $\mathcal{I}_{\text{joint}}$. It accepts $[\boldsymbol{x}_{(i-1)E}{}^T, \boldsymbol{x}_{iB}{}^T]^T \in \mathbb{R}^{26}$ as the input and produces forces and moments so as to make $\boldsymbol{x}_{(i-1)E}$ to be equal to $\boldsymbol{x}_{iB}$ except in the rotational DOF around the joint axis. Also, $\mathcal{I}_{\text{joint}}$ must take one state variable $e_i \in \mathbb{R}$ for the rotational friction around the axis. By using such an appropriate function $\mathcal{I}_{\text{joint}}$, the algorithm in one timestep can be described in the following form:

$$\text{For } r = 1,\ 2,\ 3,\ 4 \tag{29a}$$

$$t = kT + T_{r-1} \tag{29b}$$

$$\tau_{M,0} = \begin{cases} \min(0.52, 0.3t) & \text{if } t < 4 \\ 0.336 + 0.144\sin(100t) & \text{otherwise} \end{cases} \tag{29c}$$

$$\tau_{M,1} = \begin{cases} \min(0.52, 0.3t) & \text{if } t < 4 \\ 0.336 + 0.144\cos(100t) & \text{otherwise} \end{cases} \tag{29d}$$

$$e_0 = e_0(k) + \varepsilon_{0,r-1}T_{r-1} \tag{29e}$$

$$e_1 = e_1(k) + \varepsilon_{1,r-1}T_{r-1} \tag{29f}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}_0(k) + \boldsymbol{\xi}_{0,r-1}T_{r-1} \tag{29g}$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_1(k) + \boldsymbol{\xi}_{1,r-1}T_{r-1} \tag{29h}$$

$$\boldsymbol{x}_{0B} = \boldsymbol{x}_{0B}(\boldsymbol{x}_0) \tag{29i}$$

$$\boldsymbol{x}_{0E} = \boldsymbol{x}_{0E}(\boldsymbol{x}_0) \tag{29j}$$

$$\boldsymbol{x}_{1B} = \boldsymbol{x}_{1B}(\boldsymbol{x}_1) \tag{29k}$$

$$\begin{bmatrix} \boldsymbol{f}_0 \\ \boldsymbol{\tau}_0 \\ \varepsilon_{0,r} \end{bmatrix} = \mathcal{I}_{\text{joint}}\left(\begin{bmatrix} \boldsymbol{o} \\ \boldsymbol{x}_{0B} \end{bmatrix}, e_0; T_r\right) \tag{29l}$$

$$\begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{\tau}_1 \\ \varepsilon_{1,r} \end{bmatrix} = \mathcal{I}_{\text{joint}}\left(\begin{bmatrix} \boldsymbol{x}_{0E} \\ \boldsymbol{x}_{1B} \end{bmatrix}, e_1; T_r\right) \tag{29m}$$

$$\boldsymbol{\xi}_{0,i} = \mathcal{A}_{\text{rigid}}\left(\boldsymbol{x}_0, \begin{bmatrix} \boldsymbol{f}_0 - \boldsymbol{f}_1 \\ \boldsymbol{\tau}_0 - \boldsymbol{\tau}_1 + \boldsymbol{s}_{0B} \times \boldsymbol{f}_0 - \boldsymbol{s}_{0E} \times \boldsymbol{f}_1 \end{bmatrix}\right) \tag{29n}$$

$$\boldsymbol{\xi}_{1,i} = \mathcal{A}_{\text{rigid}}\left(\boldsymbol{x}_1, \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{\tau}_1 + \boldsymbol{s}_{1B} \times \boldsymbol{f}_1 \end{bmatrix}\right) \tag{29o}$$

$$\text{End For} \tag{29p}$$

$$e_0(k+1) = e_0(k) + (T/6)(\varepsilon_{0,1} + 2\varepsilon_{0,2} + 2\varepsilon_{0,3} + \varepsilon_{0,4}) \tag{29q}$$

$$e_1(k+1) = e_1(k) + (T/6)(\varepsilon_{1,1} + 2\varepsilon_{1,2} + 2\varepsilon_{1,3} + \varepsilon_{1,4}) \tag{29r}$$

$$\boldsymbol{x}_0(k+1) = \boldsymbol{x}_0(k) + (T/6)(\boldsymbol{\xi}_{0,1} + 2\boldsymbol{\xi}_{0,2} + 2\boldsymbol{\xi}_{0,3} + \boldsymbol{\xi}_{0,4}) \tag{29s}$$

$$\boldsymbol{x}_1(k+1) = \boldsymbol{x}_1(k) + (T/6)(\boldsymbol{\xi}_{1,1} + 2\boldsymbol{\xi}_{1,2} + 2\boldsymbol{\xi}_{1,3} + \boldsymbol{\xi}_{1,4}) \tag{29t}$$

Here, $\boldsymbol{s}_{iB} = \boldsymbol{p}_{iB} - \boldsymbol{p}_i$ and $\boldsymbol{s}_{iE} = \boldsymbol{p}_{iE} - \boldsymbol{p}_i$, where $\boldsymbol{p}_{iB}$ is a part of $\boldsymbol{x}_{iB}$ as in (25).

The simulation yielded almost the same result as in Fig. 7 when the stiffness of the joints are set to be high enough. An advantage of this approach is that it does not require analytical expressions of the equation of motion of the link mechanism. A disadvantage of this method is that, in this approach, the stiffness of the joint should be set high enough and the timestep size should be set small enough.

V. Conclusions

This paper has described a modular architecture of simulation software that is capable of capturing Coulomb friction in mechanical systems. The presented architecture is built upon Kikuuwe et al.'s discrete-time friction model, which is originally based on the implicit Euler method. The paper has modified the model into a form that can be integrated through the standard fourth order Runge-Kutta method. Some example implementations have been presented.

Future research will investigate further improvements of the presented model to capture complicated friction phenomena, such as hysteresis in the static friction and surface stiction in micro/nano-scale systems. Hysteresis behaviors will be realized by a combination of multiple friction components, as in [10]. The application of the technique for a more general class of nonsmooth and highly nonlinear phenomena, such as collision, will also need to be investigated.

References

[1] D. Karnopp, "Computer simulation of stick-slip friction in mechanical dynamic systems," *Trans. of ASME: J. of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 100–103, 1985.

[2] P. R. Dahl, "A solid friction model," Aerospace Corporation, Tech. Rep. TOR-0158(3107-18)-1, 1968.

[3] C. Canudas de Wit et al., "A new model for control of systems with friction," *IEEE Trans. on Automatic Control*, vol. 40, no. 3, 1995.

[4] P. Dupont et al., "Single state elastoplastic friction models," *IEEE Trans. on Automatic Control*, vol. 47, no. 5, pp. 787–792, 2002.

[5] R. Kikuuwe et al., "Admittance and impedance representations of friction based on implicit Euler integration," *IEEE Trans. on Robotics*, vol. 22, no. 6, pp. 1176–1188, 2006.

[6] T. Yoshikawa and H. Ueda, "Module-based architecture of world model for haptic virtual reality," in *Experimental Robotics V*, Springer-Verlag, 1997, pp. 155–166.

[7] K. Yamane and Y. Nakamura, "Parallel $O(\log n)$ algorithm for dynamics simulation of humanoid robots," in *Proc. of IEEE-RAS Int. Conf. on Humanoid Robotics*, 2006, pp. 554–559.

[8] R. Kikuuwe et al., "Velocity-bounding stiff position controller," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 3050–3055.

[9] M. A. Otaduy and M. C. Lin, "A modular haptic rendering algorithm for stable and transparent 6-DOF manipulation," *IEEE Trans. on Robotics*, vol. 22, no. 4, pp. 751–762, 2006.

[10] W. D. Iwan, "A distributed-element model for hysteresis and its steady-state dynamic response," *Trans. of ASME: J. of Applied Mechanics*, vol. 33, no. 4, pp. 893–900, 1966.