Analytically Differentiable Articulated Rigid Body Simulation

Simulation Document for "An End-to-End Differentiable Framework for Contact-Aware Robot Design"

Jie Xu[†] and Shinjiro Sueda[‡]
[†]Massachusetts Institute of Technology
[‡]Texas A&M University
http://diffhand.csail.mit.edu

This document includes the mathematical details of the differentiable simulation developed in the paper "*An End-to-End Differentiable Framework for Contact-Aware Robot Design*". Our simulation is based on RedMax [1], which represents the dynamic system in a reduced coordinate formulation and applies implicit time stepping as the integration scheme. A differentiable version of RedMax[1] has been implemented in MATLAB and computes the analytical gradients for control and state variables. In our work, we extend differentiable RedMax to include the analytical gradients to a full spectrum of simulation parameters including kinematics- and dynamics-related parameters. Such gradient information has not been supported by any existing differentiable simulators (to the best of our knowledge) and unlocks the opportunity to apply efficient gradient-based optimization to search for the design parameters of the robots. Furthermore, we generalize the penalty-based frictional contact model proposed by Geilinger et al. [2] that only supports contact between a single dynamic body and a stationary surface (*i.e.*, ground and walls) to support contact between multiple dynamic bodies. The core of our simulation is fully implemented in C++ and provides Python interfaces through pybind11 to allow downstream applications on Python side.

In this document, we mainly focus on the details of the extension part we made over the original differentiable RedMax including the penalty-based contact formulation between dynamic bodies and its analytical gradients, and the analytical gradients to the simulation parameters. For the basic formulation and the analytical gradients to the control and states variables, we brifly introduce them in this document and refer the reader to the full and detailed notes provided by Sueda[2].

I. Dynamics Formulation and Notations of RedMax

RedMax [1] is an efficient articulated body simulation. RedMax is known for its compact reduced coordinates representation and its flexibility for rigid-soft material coupling. In this document, we focus on its rigid body dynamics. We introduce its forward dynamics formulation in Section I-B and the implicit time integration in Section I-C.

A. Notations

To make the whole document tractable for reading, we provide a look-up notation table in Table I. Since RedMax define most of the dynamics in local joint/body coordinate, it frequently conducts conversions between coordination frames. We massively use the subscript and superscript to denote coordinate frames. Without special clarification, frame 0 denotes the static inertial frame of world space throughout this document.

B. Dynamics in Reduced Coordinates System

The dynamics equation of the RedMax is:

$$\mathsf{M}_r \ddot{\mathsf{q}}_r = \mathsf{f}_r \tag{1}$$

$$\Rightarrow \mathsf{M}_r \ddot{\mathsf{q}}_r = \tilde{\mathsf{f}}_r + \mathsf{J}_{mr}^T \mathsf{f}_m + \mathsf{f}_{QVV} + u \tag{2}$$

where $\mathsf{M}_r = \mathsf{J}_{mr}^T \mathsf{M}_m \mathsf{J}_{mr}$ and $\mathsf{f}_{QVV} = -\mathsf{J}_{mr}^T \mathsf{M}_m \dot{\mathsf{J}}_{mr} \dot{\mathsf{q}}_r$.

C. Implicit Time Integration

RedMax steps forward the simulation through implicit time integration of Equation 2. In our implementation we support both **BDF1** and **BDF2** integration schemes. Without special notation, we remove the subscript r for brevity and all symbols without subscript are represented in reduced coordinates.

[1]The Matlab code of differentiable RedMax can be found at: https://github.com/sueda/redmax
[2]The full and detailed version of the introduction of differentiable RedMax is provided by Sueda at: https://github.com/sueda/redmax/blob/master/notes.pdf

TABLE I
SIMULATION NOTATIONS

n_m	Number of rigid bodies in the system.
n_r	Number of degrees of freedom of the system.
h	Time step size in simulation.
f_m	Maximal wrench vector including force and torque generated in maximal coordinate systems such as gravity and Coriolis forces. 6D for each body.
f_r	Reduced force/torque vector defined on reduced degrees of freedom.
$\tilde{\mathsf{f}}_r$	Reduced force/torque vector generated by joint-space effect (*e.g.*, joint damping and stiffness).
f_{QVV}	Quadratic velocity vector.
u	Generalized joint torque vector generated by actuators (*e.g.*, motors).
q_m	Maximal state vector of the simulation. $\mathsf{q}_m \in \mathbb{R}^{6n_m}$. $\dot{\mathsf{q}}_m$ is the concatenation of ${}^i\phi_i$ of all rigid bodies.
q_r	Reduced state vector of the simulation. $\mathsf{q}_m \in \mathbb{R}^{n_r}$.
M_m	Maximal inertia matrix. $\mathsf{M}_m \in \mathbb{R}^{6n_m \times 6n_m}$.
M_r	Generalized inertia matrix in reduced coordinates. $\mathsf{M}_r \in \mathbb{R}^{n_r \times n_r}$
J_{mr}	Jacobian mapping generalized velocity $\dot{\mathsf{q}}_r$ to its maximal coordinate counterpart $\dot{\mathsf{q}}_m$, *i.e.*, $\dot{\mathsf{q}}_m = \mathsf{J}_{mr}\dot{\mathsf{q}}_r$. We use J for brevity in the document.

a) BDF1: Under **BDF1** integration scheme, the dynamics system is discretized into:

$$\dot{\mathsf{q}}^{(k+1)} = \dot{\mathsf{q}}^{(k)} + h\mathsf{M}^{-1}\mathsf{f}\big(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}\big) \tag{3}$$

$$\mathsf{q}^{(k+1)} = \mathsf{q}^{(k)} + h\dot{\mathsf{q}}^{(k+1)} \tag{4}$$

We can replace $\dot{\mathsf{q}}$ by $\frac{1}{h}\big(\mathsf{q}^{(k+1)} - \mathsf{q}^{k)}\big)$ in Equation 3 and rearrange it:

$$g(\mathsf{q}^{(k+1)}) = \mathsf{q}^{(k+1)} - \mathsf{q}^{(k)} - h\dot{\mathsf{q}}^{(k)} - h^2\mathsf{M}^{-1}\mathsf{f}\big(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}\big) = 0 \tag{5}$$

Newton's method is applied to solve the nonlinear Equation 5 with the availability of first-order gradients $\frac{\partial(\mathsf{M},\mathsf{f})}{\partial \mathsf{q}}$ and $\frac{\partial(\mathsf{M},\mathsf{f})}{\partial \dot{\mathsf{q}}}$.

b) BDF2: Since **BDF2** is a multi-step method, we cannot use it to start the simulation–it needs access to two previous states rather than just one. Therefore, to start **BDF2**, we use **SDIRK2**, which is:

$$\dot{\mathsf{q}}^{(\alpha)} = \dot{\mathsf{q}}^{(0)} + \alpha h\mathsf{M}^{-1}\mathsf{f}\big(\mathsf{q}^{(\alpha)}, \dot{\mathsf{q}}^{(\alpha)}\big) \tag{6}$$

$$\mathsf{q}^{(\alpha)} = \mathsf{q}^{(0)} + \alpha h\dot{\mathsf{q}}^{(\alpha)} \tag{7}$$

$$\dot{\mathsf{q}}^{(1)} = \dot{\mathsf{q}}^{(0)} + (1-\alpha)h\mathsf{M}^{-1}\mathsf{f}\big(\mathsf{q}^{(\alpha)}, \dot{\mathsf{q}}^{(\alpha)}\big) + \alpha h\mathsf{M}^{-1}\mathsf{f}\big(\mathsf{q}^{(1)}, \dot{\mathsf{q}}^{(1)}\big) \tag{8}$$

$$\mathsf{q}^{(1)} = \mathsf{q}^{(0)} + (1-\alpha)h\dot{\mathsf{q}}^{(\alpha)} + \alpha h\dot{\mathsf{q}}^{(1)}, \tag{9}$$

where $\alpha = (2-\sqrt{2})/2$.

For the steps afterwards, we apply **BDF2**:

$$\dot{\mathsf{q}}^{(k+1)} = \frac{4}{3}\dot{\mathsf{q}}^{(k)} - \frac{1}{3}\dot{\mathsf{q}}^{(k-1)} + \frac{2}{3}h\mathsf{M}^{-1}\mathsf{f}\big(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}\big) \tag{10}$$

$$\mathsf{q}^{(k+1)} = \frac{4}{3}\mathsf{q}^{(k)} - \frac{1}{3}\mathsf{q}^{(k-1)} + \frac{2}{3}h\dot{\mathsf{q}}^{(k+1)} \tag{11}$$

Same as **BDF1**, we can construct a nonlinear implicit function $g(\mathsf{q}^{(k+1)})$ by representing $\dot{\mathsf{q}}^{(k+1)}$ by $\mathsf{q}^{(k+1)}$ and rearranging, and apply Newton's method to solve $g(\mathsf{q}^{(k+1)}) = 0$.

II. SIMULATION PARAMETERS

To enable efficient optimization for robot shape design, we define a set of simulation parameters or called design parameters and compute the gradients of the simulation results with respect to those simulation parameters. The simulation parameters can usually be divided into two categories depending on whether they control the kinematic property or the dynamic property of the robot as shown in Table II. The kinematic parameters affects the kinematic tree of the robot, and includes the transformation matrices of each joint and body in the rest configuration. The dynamic parameters include the maximal inertia M_m (containing mass m and inertia tensor I), contact point locations, joint stiffness/damping coefficient, and the contact coefficient scale (affecting the contact/frictional coefficients and equivalent to the contact area mentioned in the paper). Since different parameters affect the simulation results in different ways, the gradient computation for them can be very different. Therefore, we label each type of parameters as a unique index and treat them differently during the computation in order to compute the gradient more efficiently. Furthermore, some design optimization methods may not want to optimize all of them, our implemented

TABLE II
LIST OF SIMULATION PARAMETERS

Type	Notation	Parameter Description	Dimension	Index	Bit mask
Kinematics	E_j	Joint transformation	SE(3)	I	1
	E_b	Body transformation	SE(3)	II	2
Dynamics	C_b	Contact points on body	$3 \times n_c$	III	4
	m_i	Body mass	1	IV	8
	I_i	Body diagonal inertia tensor	3	IV	8
	K_r	Joint stiffness coefficient	1	V	16
	D_r	Joint damping coefficient	1	V	16
	s	Contact coefficient scale	1	VI	32

simulation allows the user to define in the simulation configuration file what types of the simulation parameters of each rigid body needs the gradient information. This is achieved by set a binary value for each rigid body as a bit mask. Such flexibility enable a more efficient gradient computation by not wasting computation on the gradients not being used. Here we give an example of the bit mask. For a rigid body, if we set the simulation bit mask to be $11 = 1 + 2 + 8$, then it means we would like to compute the gradient for its joint transformation, body transformation, and mass inertia. To be noted, the dimension shown in the table is different from the ones shown in the paper since here we count the dimension for each rigid body.

We analytically derive the gradients of the simulation with respect to those kinematic and dynamic parameters and provide the detailed derivation of those gradients in Section V. (We have not derived the gradients for the joint stiffness/damping coefficient since we do not consider these two types of parameters in our optimization. However the gradient to these parameters is much easier to derive compared to other parameters and we will add those gradient computation as soon as possible to make the simulation feature-complete.)

III. DIFFERENTIABLE FRICTIONAL CONTACT MODEL

We generalize the penalty-based frictional contact model proposed by Geilinger, el. al. [2] that only showed contact between a single dynamic body and a static surface (*e.g.*, wall) to support more general contact cases where the contact can happen between dynamic bodies, a key requirement for object manipulation. Furthermore, we modify the contact damping force so that it becomes free of discontinuities.

Specifically, we consider the contact between a robot body component represented by a mesh with a set of contact points and a manipulated object represented by a signed distance field. Let the states (*i.e.*, generalized coordinates) of the robot and the manipulated object be q_b and q_o. While conducting collision detection, the world position and velocity of the contact point C_b on a robot body B_i are computed by the kinematic tree of the robot as $\boldsymbol{x}(\mathsf{q}_b, C_b)$ and $\dot{\boldsymbol{x}}(\mathsf{q}_b, \dot{\mathsf{q}}_b, C_b)$. Given the world position and velocity of a point, we obtain the following quantities through the signed distance function attached to the manipulated object:

$$\begin{array}{ll} d,\, \dot{d} & \text{penetration distance and speed} \\ \mathbf{n} & \text{contact normal} \\ \dot{\mathbf{t}} & \text{tangential velocity} \end{array}$$

where the contact normal determine the direction of the contact force and the tangential velocity determine the magnitude and direction of the frictional force.

Then the contact force and frictional force are computed when there exist penetration (*i.e.*, $d < 0$). The contact force applied on the robot is computed as:

$$f_c = (-k_n d + k_d d\dot{d}) s\mathbf{n} \tag{12}$$

where k_n is the contact stiffness coefficient and s is the contact coefficient scale. The first part is the penalty-based contact force while the second part is the contact damping force to help stabilize the simulation. Here, instead of using the original formulation $k_d\dot{d}$ proposed in [2], we use $k_d d\dot{d}$ as the magnitude of the damping force. That is because in the original formulation, there will be a sudden change in the magnitude of the damping force at $d = 0$, since $\dot{d}$ is not necessarily zero when d is zero, which will have convergence issues in some cases during the Newton's solver. Our small modification help keep the continuity of contact force at the moment of collision.

Its corresponding local-frame wrench applied on the robot body is:

$$\mathsf{f}_c = \mathsf{J}_c^T f_c = (-k_n d + k_d d\dot{d}) s \mathsf{J}_c^T \mathbf{n} \tag{13}$$

where J_c is the contact Jacobian of body B_i.

The wrench of frictional force is computed by:

$$\mathsf{f}_t = -\min(k_t\|\dot{\mathbf{t}}\|, \mu\|f_c\|)s\mathsf{J}_c\frac{\dot{\mathbf{t}}}{\|\dot{\mathbf{t}}\|} \tag{14}$$

where k_t is the stiffness coefficient for the frictional force and μ is the coulomb frictional coefficient. The first part of the min function can be regarded as the static frictional force and the second part is corresponding to the dynamic frictional force.

The contact force and frictional forces applied on the manipulated object are computed in the same way but in opposite directions. Besides the forces computation, we also compute the derivatives of those force quantities with respect to $\mathsf{q}_b, \mathsf{q}_o$, and simulation parameters. The derivatives of the frictional contact forces with respect to q_b and q_o are used to step the simulation forward in time with the implicit integrator, and the derivatives with respect to simulation parameters are used to calculate the sensitivities during the backward pass of the simulation as we will mention in the Section IV.

IV. Differentiable RedMax

In this section, we show how to use implicit function theory (*i.e.*, sensitivity analysis) and adjoint method to efficient compute the gradients of the simulation results w.r.t the simulation and control variables. We consider a general task objective $\Psi(\boldsymbol{p}, \bar{\mathbf{q}}, \bar{v})$, where $\boldsymbol{p} = \{\boldsymbol{p_d}, \mathbf{q}_0, \boldsymbol{u}\}$ is the optimization variables. Here $\boldsymbol{p}_d$ is the simulation parameters we introduced before including the kinematics- and dynamics-related parameters, $\mathbf{q}_0$ is the initial states of the system, and $\boldsymbol{u}$ is the control sequence (*e.g.*, joint torque sequence) of the robot. $\bar{\mathbf{q}}(\boldsymbol{p})$ is the concatenated vector of all state vectors $\mathsf{q}^{(t)}$, and $\bar{v}(\boldsymbol{p}, \bar{\mathbf{q}})$ is the auxiliary variables defined by the user such as the positions of end effectors.

A typical optimization problem can be then formulated as:

$$\underset{\boldsymbol{p}}{\text{minimize}} \;\; \Psi\big(\boldsymbol{p}, \bar{\mathbf{q}}(\boldsymbol{p}), \bar{v}(\boldsymbol{p}, \bar{\mathbf{q}})\big) \tag{15}$$

To apply a gradient-based solver (*e.g.*, L-BFGS-B) for the optimization problem defined in Eq. 15, we would like to have the following derivatives:

$$\begin{aligned}\frac{d\Psi}{d\boldsymbol{p}} &= \frac{\partial\Psi}{\partial\boldsymbol{p}} + \frac{\partial\Psi}{\partial\bar{\mathbf{q}}}\frac{d\bar{\mathbf{q}}}{d\boldsymbol{p}} + \frac{\partial\Psi}{\partial\bar{v}}\Big(\frac{\partial\bar{v}}{\partial\bar{\mathbf{q}}}\frac{d\bar{\mathbf{q}}}{d\boldsymbol{p}} + \frac{\partial\bar{v}}{\partial\boldsymbol{p}}\Big) \\ &= \frac{\partial\Psi}{\partial\boldsymbol{p}} + \Big(\frac{\partial\Psi}{\partial\bar{\mathbf{q}}} + \frac{\partial\Psi}{\partial\bar{v}}\frac{\partial\bar{v}}{\partial\bar{\mathbf{q}}}\Big)\frac{d\bar{\mathbf{q}}}{d\boldsymbol{p}} + \frac{\partial\Psi}{\partial\bar{v}}\frac{\partial\bar{v}}{\partial\boldsymbol{p}}\end{aligned} \tag{16}$$

Since we have constraints on the implicit integration function $\boldsymbol{g}(\boldsymbol{p}, \bar{\mathbf{q}}) = 0$ at each time step, we can use the implicit function theorem to compute the derivative $\frac{d\bar{\mathbf{q}}}{d\boldsymbol{p}}$:

$$\frac{d\bar{\mathbf{q}}}{d\boldsymbol{p}} = -\Big(\frac{\partial\bar{\boldsymbol{g}}}{\partial\bar{\mathbf{q}}}\Big)^{-1}\frac{\partial\bar{\boldsymbol{g}}}{\partial\boldsymbol{p}} \tag{17}$$

Then the Eq. 16 becomes:

$$\frac{d\Psi}{d\boldsymbol{p}} = \frac{\partial\Psi}{\partial\boldsymbol{p}} - \Big(\frac{\partial\Psi}{\partial\bar{\mathbf{q}}} + \frac{\partial\Psi}{\partial\bar{v}}\frac{\partial\bar{v}}{\partial\bar{\mathbf{q}}}\Big)\Big(\frac{\partial\bar{\boldsymbol{g}}}{\partial\bar{\mathbf{q}}}\Big)^{-1}\frac{\partial\bar{\boldsymbol{g}}}{\partial\boldsymbol{p}} + \frac{\partial\Psi}{\partial\bar{v}}\frac{\partial\bar{v}}{\partial\boldsymbol{p}} \tag{18}$$

We use adjoint method to save the computation time. For the middle part, $\frac{\partial\Psi}{\partial\bar{\mathbf{q}}} + \frac{\partial\Psi}{\partial\bar{v}}\frac{\partial\bar{v}}{\partial\bar{\mathbf{q}}}$ is 1-by-Tn_r, $\Big(\frac{\partial\bar{\boldsymbol{g}}}{\partial\bar{\mathbf{q}}}\Big)^{-1}$ is Tn_r-by-Tn_r, and $\frac{\partial\bar{\boldsymbol{g}}}{\partial\boldsymbol{p}}$ is Tn_r-by-n_p. Thus we can save the time complexity on computing their multiplication by grouping the first two terms:

$$\frac{d\Psi}{d\boldsymbol{p}} = \frac{\partial\Psi}{\partial\boldsymbol{p}} - \left(\Big(\frac{\partial\Psi}{\partial\bar{\mathbf{q}}} + \frac{\partial\Psi}{\partial\bar{v}}\frac{\partial\bar{v}}{\partial\bar{\mathbf{q}}}\Big)\Big(\frac{\partial\bar{\boldsymbol{g}}}{\partial\bar{\mathbf{q}}}\Big)^{-1}\right)\frac{\partial\bar{\boldsymbol{g}}}{\partial\boldsymbol{p}} + \frac{\partial\Psi}{\partial\bar{v}}\frac{\partial\bar{v}}{\partial\boldsymbol{p}} \tag{19}$$

Let $\bar{\boldsymbol{y}} = \frac{\partial\Psi}{\partial\bar{\mathbf{q}}} + \frac{\partial\Psi}{\partial\bar{v}}\frac{\partial\bar{v}}{\partial\bar{\mathbf{q}}}$ and $\bar{H} = \frac{\partial\bar{\boldsymbol{g}}}{\partial\bar{\mathbf{q}}}$. We first solve

$$\bar{H}^T\bar{\boldsymbol{z}} = \bar{\boldsymbol{y}} \tag{20}$$

The block banded structure of the hessian matrix $\bar{H}$ allows us to solve Eq. 20 efficiently with a block banded triangular solver. To compute $\hat{H}$, the Section 3.5 of Sueda's notes[3] introduces the detailed maths for it. Since $\hat{H}$ is also required by the implicit solver during forward pass, we store the computed $\hat{H}$ and its matrix factors at each time step during forward pass, and re-use those stored variables during the backward pass to save gradient computation time.

[3] https://github.com/sueda/redmax/blob/master/notes.pdf

After solve the Eq. 20, we can then form the derivative as:

$$\frac{d\Psi}{d\boldsymbol{p}} = \frac{\partial \Psi}{\partial \boldsymbol{p}} - \bar{\boldsymbol{z}}^T \frac{\partial \bar{\boldsymbol{g}}}{\partial \boldsymbol{p}} + \frac{\partial \Psi}{\partial \bar{\boldsymbol{v}}} \frac{\partial \bar{\boldsymbol{v}}}{\partial \boldsymbol{p}} \tag{21}$$

The last missing part here is how to compute $\frac{\partial \bar{\boldsymbol{g}}}{\partial \boldsymbol{p}}$. Its derivation has to be discussed for **BDF1** and **BDF2** individually.

a) BDF1:

$$g^{(k+1)} = \mathsf{M}^{(k+1)}\big(\mathsf{q}^{(k+1)} - \mathsf{q}^{(k)} - h\dot{\mathsf{q}}^{(k)}\big) - h^2 \mathsf{f}_r\big(\mathsf{q}^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}, \boldsymbol{p}\big) \tag{22}$$

$$\frac{\partial g^{(k+1)}}{\partial \boldsymbol{p}} = \frac{\partial \mathsf{M}^{(k+1)}}{\partial \boldsymbol{p}} \otimes \big(\mathsf{q}^{(k+1)} - \mathsf{q}^{(k)} - h\dot{\mathsf{q}}^{(k)}\big) - h^2 \frac{\partial \mathsf{f}_r^{(k+1)}}{\partial \boldsymbol{p}} \tag{23}$$

b) BDF2: BDF2 includes two stages for the integration, SDIRK2 and BDF2.

For SDIRK2 step:

$$g^{(\alpha)} = \mathsf{M}^{(\alpha)}\big(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)} - \alpha h\dot{\mathsf{q}}^{(0)}\big) - \alpha^2 h^2 \mathsf{f}_r^{(\alpha)}\big(\mathsf{q}^{(\alpha)}, \dot{\mathsf{q}}^{(\alpha)}, \boldsymbol{p}\big) \tag{24}$$

$$g^{(1)} = \mathsf{M}^{(1)}\big(\mathsf{q}^{(1)} - \mathsf{q}^{(0)} - (2\alpha - 1)h\dot{\mathsf{q}}^{(0)} - \frac{2(1-\alpha)}{\alpha}(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)})\big) - \alpha^2 h^2 \mathsf{f}_r\big(\mathsf{q}^{(1)}, \dot{\mathsf{q}}^{(1)}, \boldsymbol{p}\big) \tag{25}$$

$$\frac{\partial g^{(\alpha)}}{\partial \boldsymbol{p}} = \frac{\partial \mathsf{M}^{(\alpha)}}{\partial \boldsymbol{p}} \otimes \big(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)} - \alpha h\dot{\mathsf{q}}^{(0)}\big) - \alpha^2 h^2 \frac{\partial \mathsf{f}_r^{(\alpha)}}{\partial \boldsymbol{p}} \tag{26}$$

$$\frac{\partial g^{(1)}}{\partial \boldsymbol{p}} = \frac{\partial \mathsf{M}^{(1)}}{\partial \boldsymbol{p}} \otimes \big(\mathsf{q}^{(1)} - \mathsf{q}^{(0)} - (2\alpha - 1)h\dot{\mathsf{q}}^{(0)} - \frac{2(1-\alpha)}{\alpha}(\mathsf{q}^{(\alpha)} - \mathsf{q}^{(0)})\big) - \alpha^2 h^2 \frac{\partial \mathsf{f}_r^{(1)}}{\partial \boldsymbol{p}} \tag{27}$$

For BDF2 step:

$$g^{(k+1)} = \mathsf{M}^{(k+1)}\big(\mathsf{q}^{(k+1)} - \frac{4}{3}\mathsf{q}^{(k)} + \frac{1}{3}\mathsf{q}^{(k-1)} - \frac{8}{9}h\dot{\mathsf{q}}^{(k)} + \frac{2}{9}h\dot{\mathsf{q}}^{(k-1)}\big) - \frac{4}{9}h^2 \mathsf{f}_r\big(q^{(k+1)}, \dot{\mathsf{q}}^{(k+1)}, \boldsymbol{p}\big) \tag{28}$$

$$\frac{\partial g^{(k+1)}}{\partial \boldsymbol{p}} = \frac{\partial \mathsf{M}^{(k+1)}}{\partial \boldsymbol{p}} \otimes \big(\mathsf{q}^{(k+1)} - \frac{4}{3}\mathsf{q}^{(k)} + \frac{1}{3}\mathsf{q}^{(k-1)} - \frac{8}{9}h\dot{\mathsf{q}}^{(k)} + \frac{2}{9}h\dot{\mathsf{q}}^{(k-1)}\big) - \frac{4}{9}h^2 \frac{\partial \mathsf{f}_r^{(k+1)}}{\partial \boldsymbol{p}} \tag{29}$$

The computation of individual derivatives terms are detailed in Section V. By combining all the individual derivatives terms, we are now able to analytically compute $\frac{\partial \bar{\boldsymbol{g}}}{\partial \boldsymbol{p}}$, and thus $\frac{d\Psi}{d\boldsymbol{p}}$.

V. Analytical Gradients for Simulation Parameters

A. Inertia and Force Derivatives

In this section, we derive the inertia derivative $\frac{\partial \mathsf{M}}{\partial \boldsymbol{p}}$ and force derivatives $\frac{\partial \mathsf{f}_r}{\partial \boldsymbol{p}}$.

From Eq. 2, we have $\mathsf{M} = \mathsf{J}_{mr}^T \mathsf{M}_m \mathsf{J}_{mr}$, $\mathsf{f}_r = \tilde{\mathsf{f}}_r + \mathsf{J}_{mr}^T \mathsf{f}_m + \mathsf{f}_{QVV} + u$, and $\mathsf{f}_{QVV} = -\mathsf{J}_{mr}^T \mathsf{M}_m \dot{\mathsf{J}}_{mr} \dot{\mathsf{q}}_r$. In the following, we use J without any subscripts to mean J_{mr}.

1) Inertia M*:*

$$\mathsf{M} = \mathsf{J}^T \mathsf{M}_m \mathsf{J} \tag{30}$$

For design parameters of index I and II (refer to table II)

$$\frac{\partial \mathsf{M}}{\partial p_k} = \big(\frac{\partial \mathsf{J}}{\partial p_k}\big)^T \mathsf{M}_m \mathsf{J} + \mathsf{J}^T \mathsf{M}_m \frac{\partial \mathsf{J}}{\partial p_k} \tag{31}$$

For design parameters of index IV:

$$\frac{\partial \mathsf{M}}{\partial p_k} = \mathsf{J}^T \frac{\partial \mathsf{M}_m}{\partial p_k} \mathsf{J} \tag{32}$$

The following identities are used for derivatives:

$$\mathsf{M}_m = \begin{pmatrix} \mathsf{M}_1 & & & \\ & \mathsf{M}_2 & & \\ & & \ddots & \\ & & & \mathsf{M}_n \end{pmatrix} \tag{33}$$

$$\mathsf{M}_i = \begin{pmatrix} I_i & \\ & m_i I_{3\times 3} \end{pmatrix} \tag{34}$$

$$\frac{\partial \mathsf{M}_i}{\partial m_i} = \begin{pmatrix} 0 & 0 \\ 0 & I_{3\times 3} \end{pmatrix} \tag{35}$$

$\frac{\partial \mathsf{M}_i}{\partial I_i(k)}$ can be also computed easily. (36)

2) Quadratic Velocity Vector:

$$\mathsf{f}_{QVV} = -\mathsf{J}^T \mathsf{M}_m \dot{\mathsf{J}}\dot{\mathsf{q}} \tag{37}$$

For design parameters of index I and II:

$$\frac{\partial \mathsf{f}_{QVV}}{\partial p_k} = -\frac{\partial \mathsf{J}^T}{\partial p_k} \mathsf{M}_m \dot{\mathsf{J}}\dot{q} - \mathsf{J}^T \mathsf{M}_m \frac{\partial \dot{\mathsf{J}}}{\partial p_k} \dot{\mathsf{q}} \tag{38}$$

For design parameters of index IV:

$$\frac{\partial \mathsf{f}_{QVV}}{\partial p_k} = -\mathsf{J}^T \frac{\partial \mathsf{M}_m}{\partial p_k} \dot{\mathsf{J}}\dot{\mathsf{q}} \tag{39}$$

3) All Forces:

$$\frac{\partial \mathsf{f}_r}{\partial \boldsymbol{p}} = \frac{\partial \tilde{\mathsf{f}}_r}{\partial \boldsymbol{p}} + \left(\frac{\partial \mathsf{J}}{\partial \boldsymbol{p}}\right)^T \otimes \mathsf{f}_m + \mathsf{J}^T \frac{\partial \mathsf{f}_m}{\partial \boldsymbol{p}} + \frac{\partial \mathsf{f}_{QVV}}{\partial \boldsymbol{p}} + \frac{\partial u}{\partial \boldsymbol{p}} \tag{40}$$

REFERENCES

[1] Y. Wang, N. J. Weidner, M. A. Baxter, Y. Hwang, D. M. Kaufman, and S. Sueda, "REDMAX: Efficient & flexible approach for articulated dynamics," *ACM Trans. Graph.*, vol. 38, no. 4, Jul. 2019. [Online]. Available: https://doi.org/10.1145/3306346.3322952

[2] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, "Add: analytically differentiable dynamics for multi-body systems with frictional contact," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.