This document contains supplemental appendices for [1]. All the numbered equations in this document refer to [1]. The main paper [1] can be found at the project page: https://roahmlab.github.io/PHLAME.

APPENDIX A
INCORPORATING CONSTRAINTS INTO THE AGHF

This section explains how constraints are incorporated into the AGHF by adding them to the Lagrangian.

A. Constraint Lagrangian

We incorporate constraints into the AGHF by using a penalty term in the Lagrangian in a similar fashion to [2]. By adding this term, our original Lagrangian from (5) is augmented in the following way:

Definition A1. *Let k_{cons} be some large positive real number that penalizes constraint violation, and let $g_j(x)$ be the j-th inequality constraint evaluated at x. Finally, let L be the Lagrangian from* (5) *with additional terms to enforce the constraints for all $j \in \mathcal{J}$. L_{cons} is given by:*

$$L_{cons}(x, \dot{x}) = L(x, \dot{x}) + \sum_{j \in \mathcal{J}} b(g_j(x)) \tag{A1}$$

where

$$b(g_j(x)) = k_{cons} \cdot (g_j(x))^2 \cdot S(g_j(x)), \tag{A2}$$

where $S : \mathbb{R} \to \mathbb{R}$ is defined as follows:

1) *$S : \mathbb{R} \to \mathbb{R}$ is a positive, differentiable function,*
2) *$S(g_j(x)) = 0$ when $g_j(x) \leq 0$ and*
3) *$S(g_j(x)) = 1$ when $g_j(x) > 0$.*

An example of an S satisfying this definition is:

$$S(g_j(x)) = \frac{1}{2} + \frac{1}{2} \tanh(c_{\text{cons}} \cdot g_j(x)) \tag{A3}$$

where c_{cons} is a hyper-parameter that determines how fast $S(g_j(x))$ transitions from 0 to 1, once the constraint is violated.

We introduce the form of the AGHF with L_{cons} in the following definition:

Definition A2. *Let L_{cons} be used as the Lagrangian to construct the AGHF PDE. Then the constrained AGHF PDE is given by:*

$$\frac{\partial x}{\partial s} = G^{-1}(x) \left(\frac{d}{dt} \frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} - \sum_{j \in \mathcal{J}} \frac{\partial b(g_j(x))}{\partial x} \right) \tag{A4}$$

The proof of convergence of the constrained AGHF is given in the proof of [2, Lemma 4.1].

APPENDIX B
SOLVING THE AGHF RAPIDLY FOR HIGH DIMENSIONAL SYSTEMS

The computationally intensive part of the AGHF method is solving (7), which is a parabolic PDE. In contrast to traditional PDEs used for optimal control (e.g., the Hamilton-Jacobi-Bellman PDE), the AGHF PDE has a complexity that scales polynomially with increasing state dimension rather than exponentially. The favorable scaling properties of the AGHF are owed to the fact that the domain of (7) is always **two-dimensional** and the dimension of the range of the function scales **linearly** with the state dimension. However, evolving the AGHF quickly demands being able to evaluate the right hand side of (7) rapidly. This can be difficult for high dimensional systems as the system dynamics and its derivatives must be evaluated each time the AGHF is called. This appendix describes how our method for solving the AGHF allows for evaluating the right hand side of (7) rapidly.

A. Computing AGHF PDE Partial Derivatives Analytically

This subsection derives analytical expressions that can be used to evaluate (7) in terms of the rigid body dynamics equation (1), and how to leverage spatial vector algebra to rapidly compute these expressions. We summarize the relevant results in the following theorem.

Note, for succinctness, we have dropped the dependence on (t, s) for the following equations (i.e. $x(t, s)$ is denoted by x). Additionally, in similar fashion to the notation introduced in Section II of [1], we denote the first N and last N components of x as x_{P1} and x_{P2}, respectively. Lastly, we also drop the dependence on the x terms for the dynamics functions (i.e., $H(x_{P1})$ is denoted by just H and so on)

Theorem B3. *Consider a system with dynamics as in* (1)*. The AGHF PDE* (7) *using the G described in Lemma 4 can be written as follows:*

$$\frac{\partial x}{\partial s} = \Omega\Big(x, \dot{x}, \ddot{x}, k\Big) = \Omega_1 - (\Omega_2 - \Omega_3 + \Omega_4), \tag{B5}$$

where

$$\Omega_1 = 2\begin{bmatrix} \ddot{x}_{P1} - \dot{x}_{P2} \\ (H^TH)^{-1}\big((\dot{H}^TH + H^T\dot{H})\dot{x}_{P2} + H^TH\ddot{x}_{P2} + \dot{H}^TC + H^T\dot{C}\big) \end{bmatrix} \tag{B6}$$

$$\Omega_2 = \begin{bmatrix} -\frac{1}{k} I_{N\times N} \frac{\partial F D_0^T}{\partial x_{P1}} H^T\Big(2C + 2H\dot{x}_{P2}\Big) \\ -(H^TH)^{-1}\frac{\partial F D_0^T}{\partial x_{P2}} H^T\Big(2C + 2H\dot{x}_{P2}\Big)\Big) \end{bmatrix} \tag{B7}$$

$$\Omega_3 = \begin{bmatrix} \boldsymbol{0} \\ 2k(H^TH)^{-1}(\dot{x}_{P1} - x_{P2}) \end{bmatrix} \tag{B8}$$

$$\Omega_4 = \begin{bmatrix} 2\frac{1}{k}\left[\frac{\partial H}{\partial x_{P1}}(\dot{x}_{P2} - FD_0)\right]^T H\,(\dot{x}_{P2} - FD_0) \\ \boldsymbol{0} \end{bmatrix}, \tag{B9}$$

and $FD_0 = -H^{-1}C$.

Proof: The Affine Geometric Heat Flow (AGHF) Equation is given by

$$\frac{\partial x}{\partial s}(t,s) = G^{-1}(x(t,s))\left(\frac{d}{dt}\frac{\partial L}{\partial \dot{x}_s}(x_s(t), \dot{x}_s(t)) - \frac{\partial L}{\partial x_s}(x_s(t), \dot{x}_s(t))\right) \tag{B10}$$

Let

$$L(x_s(t), \dot{x}_s(t)) = (\dot{x}_s(t) - F_d(x_s(t)))^T G(x_s(t))\,(\dot{x}_s(t) - F_d(x_s(t))) \tag{B11}$$

where

$$\begin{aligned} G(x_s(t)) &= (\bar{F}(x_s(t))^{-1})^T K \bar{F}(x_s(t))^{-1} \\ G(x_s(t)) &= \underbrace{\begin{bmatrix} I_{N\times(2N-m)} & 0_{N\times m} \\ 0_{N\times(2N-m)} & B^{-1}H(x_{P1}(t,s)) \end{bmatrix}^T}_{(\bar{F}(x_s(t))^{-1})^T} \underbrace{\begin{bmatrix} kI_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & I_{N\times N} \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} I_{N\times(2N-m)} & 0_{N\times m} \\ 0_{N\times(2N-m)} & B^{-1}H(x_{P1}(t,s)) \end{bmatrix}}_{(\bar{F}(x_s(t))^{-1})} \end{aligned} \tag{B12}$$

We assume, without loss of generality (WLOG), that $B = I$. Under this assumption, multiplying the matrices in (B12) yields

$$G(x_s(t)) = \begin{bmatrix} kI_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & H^TH \end{bmatrix} \tag{B13}$$

From here for conciseness we drop the dependence on (t,s) for the following equations (i.e. $x(t,s)$ is denoted by x). We also drop the dependence on the x terms for the dynamics functions (i.e. $H(x_{P1})$ is denoted by just H and so on).

To compute (B10), we need to compute the derivatives $\frac{d}{dt}\frac{\partial L}{\partial \dot{x}}$ and $\frac{\partial L}{\partial x}$. We begin by showing that $\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} = \Omega_1$. First we must compute the derivative $\frac{\partial L}{\partial \dot{x}}$,

$$\frac{\partial L}{\partial \dot{x}} = 2G(\dot{x} - F_d) = 2\underbrace{\begin{bmatrix} kI_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & H^TH \end{bmatrix}}_{G}\left(\underbrace{\begin{bmatrix} \dot{x}_{P1} \\ \dot{x}_{P2} \end{bmatrix}}_{\dot{x}} - \underbrace{\begin{bmatrix} x_{P2} \\ -H^{-1}C \end{bmatrix}}_{F_d}\right) = 2\begin{bmatrix} k(\dot{x}_{P1} - x_{P2}) \\ H^TH\dot{x}_{P2} + H^TC \end{bmatrix} \tag{B14}$$

taking the time derivative of this yields:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} = 2\begin{bmatrix} k(\ddot{x}_{P1} - \dot{x}_{P2}) \\ (\dot{H}^TH + H^T\dot{H})\dot{x}_{P2} + H^TH\ddot{x}_{P2} + \dot{H}^TC + H^T\dot{C} \end{bmatrix} \tag{B15}$$

and multiplying by G^{-1} yields

$$\Omega_1 = G^{-1}\left(\frac{d}{dt}\frac{\partial L}{\partial \dot{x}}\right) = 2\begin{bmatrix} \ddot{x}_{P1} - \dot{x}_{P2} \\ (H^TH)^{-1}\big((\dot{H}^TH + H^T\dot{H})\dot{x}_{P2} + H^TH\ddot{x}_{P2} + \dot{H}^TC + H^T\dot{C}\big)\end{bmatrix} \tag{B16}$$

Next we show that $G^{-1}\frac{\partial L}{\partial x} = \Omega_2 - \Omega_3 + \Omega_4$

First, taking the derivative of (B11) wrt. x, and taking transposes to ensure that the resultant derivatives are column vectors we obtain:

$$\frac{\partial L}{\partial x} = -\frac{\partial F_d^T}{\partial x}G(\dot{x} - F_d) + (\dot{x} - F_d)^T\frac{\partial G}{\partial x}(\dot{x} - F_d) + \left((\dot{x} - F_d)^TG(-\frac{\partial F_d}{\partial x})\right)^T \tag{B17}$$

expanding the first and third terms yields

$$\frac{\partial L}{\partial x} = -\frac{\partial F_d^T}{\partial x}G\dot{x} + \frac{\partial F_d^T}{\partial x}GF_d + (\dot{x} - F_d)^T\frac{\partial G}{\partial x}(\dot{x} - F_d) + \left(-\dot{x}^TG\frac{\partial F_d}{\partial x}\right)^T + \left(F_d^TG\frac{\partial F_d}{\partial x}\right)^T \tag{B18}$$

Since G is symmetric (because kI and H^TH are symmetric) the last two terms can be simplified yielding:

$$\frac{\partial L}{\partial x} = -\frac{\partial F_d^T}{\partial x}G\dot{x} + \frac{\partial F_d^T}{\partial x}GF_d + (\dot{x} - F_d)^T\frac{\partial G}{\partial x}(\dot{x} - F_d) - \frac{\partial F_d^T}{\partial x}G\dot{x} + \frac{\partial F_d^T}{\partial x}GF_d \tag{B19}$$

Grouping terms and simplifying yields

$$\frac{\partial L}{\partial x} = 2\frac{\partial F_d^T}{\partial x}GF_d - 2\frac{\partial F_d^T}{\partial x}G\dot{x} + (\dot{x} - F_d)^T\frac{\partial G}{\partial x}(\dot{x} - F_d) \tag{B20}$$

Let $FD_0 = -H^{-1}C$. Next we derive the constituent terms in (B20). The first term is given by:

$$2\frac{\partial F_d^T}{\partial x}GF_d = 2\underbrace{\begin{bmatrix} 0_{N\times N} & \frac{\partial FD_0^T}{\partial x_{P1}} \\ I_{N\times N} & \frac{\partial FD_0^T}{\partial x_{P2}}\end{bmatrix}}_{\frac{\partial F_d^T}{\partial x}}\underbrace{\begin{bmatrix} kI_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & H^TH\end{bmatrix}}_{G}\underbrace{\begin{bmatrix} x_{P2} \\ -H^{-1}C\end{bmatrix}}_{F_d} = 2\begin{bmatrix} -\frac{\partial FD_0^T}{\partial x_{P1}}H^TC \\ kx_{P2} - \frac{\partial FD_0^T}{\partial x_{P2}}H^TC\end{bmatrix} \tag{B21}$$

and the second term:

$$2\frac{\partial F_d^T}{\partial x}G\dot{x} = 2\underbrace{\begin{bmatrix} 0_{N\times N} & \frac{\partial FD_0^T}{\partial x_{P1}} \\ I_{N\times N} & \frac{\partial FD_0^T}{\partial x_{P2}}\end{bmatrix}}_{\frac{\partial F_d^T}{\partial x}}\underbrace{\begin{bmatrix} kI_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & H^TH\end{bmatrix}}_{G}\underbrace{\begin{bmatrix} \dot{x}_{P1} \\ \dot{x}_{P2}\end{bmatrix}}_{\dot{x}} = 2\begin{bmatrix} \frac{\partial FD_0^T}{\partial x_{P1}}H^TH\dot{x}_{P2} \\ k\dot{x}_{P1} + \frac{\partial FD_0^T}{\partial x_{P2}}H^TH\dot{x}_{P2}\end{bmatrix} \tag{B22}$$

Collecting and rearranging the first 2 terms of (B20) yields:

$$2\frac{\partial F_d^T}{\partial x}GF_d - 2\frac{\partial F_d^T}{\partial x}G\dot{x} = \begin{bmatrix} -\frac{\partial FD_0^T}{\partial x_{P1}}H^T\Big(2C + 2H\dot{x}_{P2}\Big) \\ -\frac{\partial FD_0^T}{\partial x_{P2}}H^T\Big(2C + 2H\dot{x}_{P2}\Big)\end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ 2k(\dot{x}_{P1} - x_{P2})\end{bmatrix} \tag{B23}$$

and multiplying them by G^{-1} gives us Ω_2 and Ω_3:

$$G^{-1}\left(\frac{\partial F_d^T}{\partial x}GF_d + \left(F_d^TG\frac{\partial F_d}{\partial x}\right)^T - 2\frac{\partial F_d^T}{\partial x}G\dot{x}\right) = \underbrace{\begin{bmatrix} -\frac{1}{k}I_{N\times N}\frac{\partial FD_0^T}{\partial x_{P1}}H^T\Big(2C + 2H\dot{x}_{P2}\Big) \\ -(H^TH)^{-1}\frac{\partial FD_0^T}{\partial x_{P2}}H^T\Big(2C + 2H\dot{x}_{P2}\Big)\Big)\end{bmatrix}}_{\Omega_2} - \underbrace{\begin{bmatrix} \mathbf{0} \\ 2k(H^TH)^{-1}(\dot{x}_{P1} - x_{P2})\end{bmatrix}}_{\Omega_3} \tag{B24}$$

The third term of (B20) is given by:

$$(\dot{x}-F_d)^T\frac{\partial G}{\partial x}(\dot{x}-F_d)=(\dot{x}-F_d)^T\begin{bmatrix}0_{N\times N\times 2N} & \\ & \frac{\partial(H^TH)}{\partial x}\end{bmatrix}(\dot{x}-F_d)\,, \tag{B25}$$

where $\frac{\partial(H^TH)}{\partial x_i}$ be the partial derivative of H^TH wrt. the ith element of the $2N\times 1$ column vector x. Doing the vector-tensor-vector multiplication in (B25) yields an $2N\times 1$ column vector of the following form:

$$(\dot{x}-F_d)^T\frac{\partial G}{\partial x}(\dot{x}-F_d)=\begin{bmatrix}(\dot{x}-F_d)^T\begin{bmatrix}0_{N\times N} & \\ & \frac{\partial(H^TH)}{\partial x_1}\end{bmatrix}(\dot{x}-F_d)\\ \vdots \\ (\dot{x}-F_d)^T\begin{bmatrix}0_{N\times N} & \\ & \frac{\partial(H^TH)}{\partial x_{2N}}\end{bmatrix}(\dot{x}-F_d)\end{bmatrix} \tag{B26}$$

and applying (3) to (B26) yields:

$$(\dot{x}-F_d)^T\frac{\partial G}{\partial x}(\dot{x}-F_d)=\begin{bmatrix}(\dot{x}_{P2}-FD_0)^T\frac{\partial(H^TH)}{\partial x_1}(\dot{x}_{P2}-FD_0)\\ \vdots \\ (\dot{x}_{P2}-FD_0)^T\frac{\partial(H^TH)}{\partial x_{2N}}(\dot{x}_{P2}-FD_0)\end{bmatrix} \tag{B27}$$

Notice that since H is only a function of x_{P1}, then

$$\begin{bmatrix}\frac{\partial(H^TH)}{\partial x_{N+1}}\\ \vdots \\ \frac{\partial(H^TH)}{\partial x_{2N}}\end{bmatrix}=\begin{bmatrix}0\\ \vdots \\ 0\end{bmatrix} \tag{B28}$$

and

$$(\dot{x}-F_d)^T\frac{\partial G}{\partial x}(\dot{x}-F_d)=\begin{bmatrix}\begin{bmatrix}(\dot{x}_{P2}-FD_0)^T\frac{\partial(H^TH)}{\partial x_1}(\dot{x}_{P2}-FD_0)\\ \vdots \\ (\dot{x}_{P2}-FD_0)^T\frac{\partial(H^TH)}{\partial x_N}(\dot{x}_{P2}-FD_0)\end{bmatrix}\\ \begin{bmatrix}0\\ \vdots \\ 0\end{bmatrix}\end{bmatrix}=\begin{bmatrix}(\dot{x}_{P2}-FD_0)^T\frac{\partial(H^TH)}{\partial x_{P1}}(\dot{x}_{P2}-FD_0)\\ \mathbf{0}\end{bmatrix} \tag{B29}$$

Next we show how to further simplify (B29) to yield Ω_4. First, let β_i be defined as the ith entry of $(\dot{x}-F_d)^T\frac{\partial G}{\partial x}(\dot{x}-F_d)$ given by:

$$\beta_i=(\dot{x}_{P2}-FD_0)^T\frac{\partial(H^TH)}{\partial x_i}(\dot{x}_{P2}-FD_0) \tag{B30}$$

evaluating $\frac{\partial(H^TH)}{\partial x_i}$ and expanding (B30) gives:

$$\beta_i=(\dot{x}_{P2}-FD_0)^T\frac{\partial H^T}{\partial x_i}H(\dot{x}_{P2}-FD_0)+(\dot{x}_{P2}-FD_0)^TH^T\frac{\partial H}{\partial x_i}(\dot{x}_{P2}-FD_0) \tag{B31}$$

Notice β_i is a scalar, and so are each of its constituent terms. Therefore the second term of β_i can be expressed in the following way

$$\begin{aligned}(\dot{x}_{P2} - FD_0)^T H^T \frac{\partial H}{\partial x_i}(\dot{x}_{P2} - FD_0) &= \left((\dot{x}_{P2} - FD_0)^T H^T \frac{\partial H}{\partial x_i}(\dot{x}_{P2} - FD_0) \right)^T \\ &= (\dot{x}_{P2} - FD_0)^T \frac{\partial H^T}{\partial x_i} H (\dot{x}_{P2} - FD_0)\end{aligned} \tag{B32}$$

Using (B32) to simplify β_i gives

$$\beta_i = 2(\dot{x}_{P2} - FD_0)^T \frac{\partial H^T}{\partial x_i} H (\dot{x}_{P2} - FD_0) = 2\left[\frac{\partial H}{\partial x_i}(\dot{x}_{P2} - FD_0)\right]^T H (\dot{x}_{P2} - FD_0) \tag{B33}$$

Applying this to (B29) yields

$$(\dot{x} - F_d)^T \frac{\partial G}{\partial x}(\dot{x} - F_d) = \begin{bmatrix} 2\left[\frac{\partial H}{\partial x_{P1}}(\dot{x}_{P2} - FD_0)\right]^T H (\dot{x}_{P2} - FD_0) \\ \mathbf{0} \end{bmatrix} \tag{B34}$$

Note that to ensure that (B34) evaluates to the same terms as (B29) the 3rd dimension of the tensor $\frac{\partial H}{\partial x_{P1}}$ must be used in the tensor-vector multiplication with $\dot{x}_{P2} - FD_0$.

Lastly, multiplying (B34) by G^{-1} gives us Ω_4:

$$G^{-1}(\dot{x} - F_d)^T \frac{\partial G}{\partial x}(\dot{x} - F_d) = \begin{bmatrix} 2\frac{1}{k}\left[\frac{\partial H}{\partial x_{P1}}(\dot{x}_{P2} - FD_0)\right]^T H (\dot{x}_{P2} - FD_0) \\ \mathbf{0} \end{bmatrix} = \Omega_4 \tag{B35}$$

Combining these terms we have

$$\frac{\partial x}{\partial s} = G^{-1}\left(\frac{d}{dt}\frac{\partial L}{\partial \dot{x}_s} - \frac{\partial L}{\partial x_s}\right) = \Omega_1 - (\Omega_2 - \Omega_3 + \Omega_4) = \Omega\Big(x, \dot{x}, \ddot{x}, k\Big) \tag{B36}$$

■

Note that $\frac{\partial H}{\partial x_{P1}}$ is a $N \times N \times N$ tensor, so computing Ω_4 requires a matrix-tensor multiplication. Algorithm B1 provides an efficient approach to avoid explicitly constructing the tensor $\frac{\partial H}{\partial x_{P1}}$ and performing matrix-tensor multiplication directly. It also details how to efficiently evaluate the analytical expressions in Theorem B3 using spatial vector algebra and rigid body dynamics algorithms. Appendix B-C discusses the computational efficiency of these algorithms and the overall AGHF evaluation.

B. Computing the PHLAME Jacobian Partial Derivatives Analytically

To solve the system of ODEs in (22), we leverage a differential equation solver that uses an implicit method [3]. This implicit method requires the derivative of (23). Most ODE solvers that use implicit methods approximate the derivative numerically. This can reduce accuracy and increase the number of function evaluations, slowing down the process. To avoid this and further speed up PHLAME, we compute and provide the analytical Jacobian of the system of ODEs with respect to $\xi(s)$. Computing this Jacobian, requires one to compute the Jacobian of $\frac{d\xi_i}{ds}(s)$ with respect to ξ_i. We summarize the form of this Jacobian as a function of the first- and second-order derivatives of the rigid body dynamics in Theorem B4. Note that in this re-statement of the theorem first presented in [1], **we state in more detail** all the dynamics terms needed to compute the Jacobian, to make the subsequent proof clearer. To efficiently compute the required second-order derivatives, we leverage some of the algorithms highlighted in [4]. Algorithm B2 shows how we rapidly compute all the necessary terms to evaluate the Jacobian. Once again, for notational convenience we have abused notation and left out the transpose for $\frac{d\xi_i}{ds}(s)$.

Theorem B4. *Let $\xi_i(s) = x^T(t_i, s)$, $[D\xi]_i(s) = \dot{x}^T(t_i, s)$ and $[D^2\xi]_i(s) = \ddot{x}^T(t_i, s)$. Then the Jacobian of $\frac{d\xi_i}{ds}(s)$ with respect to $\xi_i(s)$, $J_{\Xi_i}(s)$ is given by:*

$$\begin{aligned}J_{\Xi_i} &= \frac{d\big(\frac{d\xi_i}{ds}(s)\big)}{d\xi_i(s)} = \frac{d\Omega\big(\xi_i^T(s), [D\xi]_i^T(s), [D^2\xi]_i^T(s), k\big)}{d\xi_i(s)} \\ &= \frac{d\Omega}{d\xi_i}\frac{d\xi_i}{d\xi_i} + \frac{d\Omega}{d[D\xi]_i}\frac{d[D\xi]_i}{d\xi_i} + \frac{d\Omega}{d[D^2\xi]_i}\frac{d[D^2\xi]_i}{d\xi_i}\end{aligned} \tag{B37}$$

where

$$\frac{d\Omega}{d\xi_i} = \frac{d\Omega_1}{dx} - (\frac{d\Omega_2}{dx} - \frac{d\Omega_3}{dx} + \frac{d\Omega_4}{dx})$$
$$= \frac{d\Omega}{d\xi_i}\left(H, \dot{H}, C, \dot{C}, FD_0, \frac{\partial H}{\partial x_{P1}}, \frac{\partial^2 H}{\partial x_{P1}^2}, \frac{\partial \dot{H}}{\partial x_{P1}}, \frac{\partial FD_0}{\partial x_{P1}}, \frac{\partial FD_0}{\partial x_{P2}}, \frac{\partial C}{\partial x_{P1}}, \frac{\partial C}{\partial x_{P2}}, \frac{\partial \dot{C}}{\partial x_{P1}}, \frac{\partial \dot{C}}{\partial x_{P2}}, \frac{\partial^2 FD_0}{\partial x_{P1}^2}, \frac{\partial^2 FD_0}{\partial x_{P2}^2}, \frac{\partial^2 FD_0}{\partial x_{P1}\partial x_{P2}}\right) \tag{B38}$$

$$\frac{d\Omega}{d[D\xi]_i} = \frac{d\Omega_1}{d\dot{x}} - (\frac{d\Omega_2}{d\dot{x}} - \frac{d\Omega_3}{d\dot{x}} + \frac{d\Omega_4}{d\dot{x}})$$
$$= \frac{d\Omega}{d[D\xi]_i}\left(H, C, \dot{H}, FD_0, \frac{\partial H}{\partial x_{P1}}, \frac{\partial FD_0}{\partial x_{P1}}, \frac{\partial FD_0}{\partial x_{P2}}, \frac{\partial C}{\partial x_{P1}}, \frac{\partial C}{\partial x_{P2}}\right) \tag{B39}$$

$$\frac{d\Omega}{d[D^2\xi]_i} = 2I_{2N\times 2N} \tag{B40}$$

Proof: Let $J_{\Xi_i}(s)$ be the Jacobian of $\frac{d\xi_i}{ds}(s)$ with respect to $\xi_i(s)$ given by:

$$J_{\Xi_i} = \frac{d\left(\frac{d\xi_i}{ds}(s)\right)}{d\xi_i(s)} = \frac{d\Omega\left(\xi_i(s), [D\xi]_i(s), [D^2\xi]_i(s), k\right)}{d\xi_i(s)}. \tag{B41}$$

Applying the chain rule for multivariable functions we obtain

$$J_{\Xi_i} = \frac{d\Omega}{d\xi_i}\frac{d\xi_i}{d\xi_i} + \frac{d\Omega}{d[D\xi]_i}\frac{d[D\xi]_i}{d\xi_i} + \frac{d\Omega}{d[D^2\xi]_i}\frac{d[D^2\xi]_i}{d\xi_i} \tag{B42}$$

From Theorem B3 we have that

$$\frac{\partial x}{\partial s} = \Omega\left(x, \dot{x}, \ddot{x}, k\right) = \Omega_1 - (\Omega_2 - \Omega_3 + \Omega_4), \tag{B43}$$

And recall that $\xi_i(s) = x^T(t_i, s)$. Using these two results, we have that

$$\frac{d\Omega}{d\xi_i} = \frac{d\Omega}{dx(t_i, s)} = \frac{d\Omega_1}{dx} - (\frac{d\Omega_2}{dx} - \frac{d\Omega_3}{dx} + \frac{d\Omega_4}{dx}) \tag{B44}$$

Computing the derivatives of each of the Ω terms yields the following:

$$\frac{d\Omega_1}{dx} = 2\begin{bmatrix} 0_{N\times N} & 0_{N\times N} \\ \frac{\partial (H^TH)^{-1}}{\partial x_{P1}}\gamma + (H^TH)^{-1}\frac{\partial \gamma}{\partial x_{P1}} & (H^TH)^{-1}\frac{\partial \gamma}{\partial x_{P2}} \end{bmatrix} \tag{B45}$$

$$\frac{d\Omega_2}{dx} = \begin{bmatrix} \frac{1}{k}I_{N\times N}\frac{\partial \alpha_1}{\partial x_{P1}} & \frac{1}{k}I_{N\times N}\frac{\partial \alpha_1}{\partial x_{P2}} \\ \frac{\partial (H^TH)^{-1}}{\partial x_{P1}}\alpha_2 + (H^TH)^{-1}\frac{\partial \alpha_2}{\partial x_{P1}} & (H^TH)^{-1}\frac{\partial \alpha_2}{\partial x_{P2}} \end{bmatrix} \tag{B46}$$

$$\frac{d\Omega_3}{dx} = 2k\begin{bmatrix} 0_{N\times N} & 0_{N\times N} \\ \frac{\partial (H^TH)^{-1}}{\partial x_{P1}}(\dot{x}_{P1} - x_{P2}) & -(H^TH)^{-1}I_{N\times N} \end{bmatrix} \tag{B47}$$

$$\frac{d\Omega_4}{dx} = \begin{bmatrix} \frac{2}{k}I_{N\times N}\frac{\partial \Gamma}{\partial x_{P1}} & \frac{2}{k}I_{N\times N}\frac{\partial \Gamma}{\partial x_{P2}} \\ 0_{N\times N} & 0_{N\times N} \end{bmatrix} \tag{B48}$$

where,

$$\gamma = (\dot{H}^TH + H^T\dot{H})\dot{x}_{P2} + H^TH\ddot{x}_{P2} + \dot{H}^TC + H^T\dot{C} \tag{B49}$$

$$\alpha_1 = -\frac{\partial FD_0^T}{\partial x_{P1}}H^T\left(2C + 2H\dot{x}_{P2}\right) \tag{B50}$$

$$\alpha_2 = -\frac{\partial FD_0^T}{\partial x_{P2}}H^T\left(2C + 2H\dot{x}_{P2}\right) \tag{B51}$$

$$\Gamma = (\dot{x}_{P2} - FD_0)^T \frac{\partial H^T}{\partial x_{P1}} H(\dot{x}_{P2} - FD_0) \tag{B52}$$

$$\begin{aligned}\frac{\partial \gamma}{\partial x_{P1}} = \frac{\partial \dot{H}^T}{\partial x_{P1}}\Big(H\dot{x}_{P2} + C\Big) + \dot{H}^T\Big(\frac{\partial H}{\partial x_{P1}}\dot{x}_{P2} + \frac{\partial C}{\partial x_{P1}}\Big) + \frac{\partial H^T}{\partial x_{P1}}\Big(\dot{H}\dot{x}_{P2} + H\ddot{x}_{P2} + \dot{C}\Big) \\ + H^T\Big(\frac{\partial \dot{H}}{\partial x_{P1}}\dot{x}_{P2} + \frac{\partial H}{\partial x_{P1}}\ddot{x}_{P2} + \frac{\partial \dot{C}}{\partial x_{P1}}\Big)\end{aligned} \tag{B53}$$

$$\frac{\partial \gamma}{\partial x_{P2}} = \dot{H}^T \frac{\partial C}{\partial x_{P2}} + H^T \frac{\partial \dot{C}}{\partial x_{P2}} \tag{B54}$$

$$\frac{\partial \alpha_1}{\partial x_{P1}} = \frac{d}{dx_{P1}}\Big(-\frac{\partial FD_0^T}{\partial x_{P1}}\Big)H^T\Big(2C + 2H\dot{x}_{P2}\Big) - \frac{\partial FD_0^T}{\partial x_{P1}}\Big(\frac{\partial H^T}{\partial x_{P1}}\Big(2C + 2H\dot{x}_{P2}\Big) + H\Big(2\frac{\partial C}{\partial x_{P1}} + 2\frac{\partial H}{\partial x_{P1}}\dot{x}_{P2}\Big)\Big) \tag{B55}$$

$$\frac{\partial \alpha_2}{\partial x_{P1}} = \frac{d}{dx_{P1}}\Big(-\frac{\partial FD_0^T}{\partial x_{P2}}\Big)H^T\Big(2C + 2H\dot{x}_{P2}\Big) - \frac{\partial FD_0^T}{\partial x_{P2}}\Big(\frac{\partial H^T}{\partial x_{P1}}\Big(2C + 2H\dot{x}_{P2}\Big) + H\Big(2\frac{\partial C}{\partial x_{P1}} + 2\frac{\partial H}{\partial x_{P1}}\dot{x}_{P2}\Big)\Big) \tag{B56}$$

$$\frac{\partial \alpha_1}{\partial x_{P2}} = \frac{d}{dx_{P2}}\Big(-\frac{\partial FD_0^T}{\partial x_{P1}}\Big)H^T\Big(2C + 2H\dot{x}_{P2}\Big) - \frac{\partial FD_0^T}{\partial x_{P1}}2H\frac{\partial C}{\partial x_{P2}} \tag{B57}$$

$$\frac{\partial \alpha_2}{\partial x_{P2}} = \frac{d}{dx_{P2}}\Big(-\frac{\partial FD_0^T}{\partial x_{P2}}\Big)H^T\Big(2C + 2H\dot{x}_{P2}\Big) - \frac{\partial FD_0^T}{\partial x_{P2}}2H\frac{\partial C}{\partial x_{P2}} \tag{B58}$$

$$\begin{aligned}\frac{\partial \Gamma}{\partial x_{P1}} = \Big(-\frac{\partial FD_0^T}{\partial x_{P1}}\frac{\partial H}{\partial x_{P1}} + (\dot{x}_{P2} - FD_0)^T\frac{\partial^2 H}{\partial x_{P1}^2}\Big)H(\dot{x}_{P2} - FD_0) \\ + (\dot{x}_{P2} - FD_0)^T\frac{\partial H}{\partial x_{P1}}\Big(\frac{\partial H}{\partial x_{P1}}(\dot{x}_{P2} - FD_0) - H\frac{\partial FD_0}{\partial x_{P1}}\Big)\end{aligned} \tag{B59}$$

$$\frac{\partial \Gamma}{\partial x_{P2}} = -\frac{\partial FD_0^T}{\partial x_{P2}}\frac{\partial H}{\partial x_{P1}}H(\dot{x}_{P2} - FD_0) + (\dot{x}_{P2} - FD_0)^T\frac{\partial H}{\partial x_{P1}}\Big(-H\frac{\partial FD_0}{\partial x_{P2}}\Big) \tag{B60}$$

The derivative of Ω wrt $[D\xi]_i$ similarly is given by

$$\frac{d\Omega}{d[D\xi]_i} = \frac{d\Omega_1}{d\dot{x}} - (\frac{d\Omega_2}{d\dot{x}} - \frac{d\Omega_3}{d\dot{x}} + \frac{d\Omega_4}{d\dot{x}}) \tag{B61}$$

$$\frac{d\Omega_1}{d\dot{x}} = 2\begin{bmatrix} 0_{N\times N} & -I_{N\times N} \\ (H^TH)^{-1}\frac{\partial \gamma}{\partial \dot{x}_{P1}} & (H^TH)^{-1}\frac{\partial \gamma}{\partial \dot{x}_{P2}}\end{bmatrix} \tag{B62}$$

$$\frac{d\Omega_2}{d\dot{x}} = \begin{bmatrix} 0_{N\times N} & \frac{1}{k}I_{N\times N}\frac{\partial \alpha_1}{\partial \dot{x}_{P2}} \\ 0_{N\times N} & (H^TH)^{-1}\frac{\partial \alpha_2}{\partial \dot{x}_{P2}})\end{bmatrix} \tag{B63}$$

$$\frac{d\Omega_3}{d\dot{x}} = \begin{bmatrix} 0_{N\times N} & 0_{N\times N} \\ 2k(H^TH)^{-1}I_{N\times N} & 0_{N\times N}\end{bmatrix} \tag{B64}$$

$$\frac{d\Omega_4}{d\dot{x}} = \begin{bmatrix} 0_{N\times N} & \frac{2}{k}I_{N\times N}\frac{\partial \Gamma}{\partial \dot{x}_{P2}} \\ 0_{N\times N} & 0_{N\times N}\end{bmatrix} \tag{B65}$$

where

$$\frac{\partial \alpha_1}{\partial \dot{x}_{P2}} = -\frac{\partial F D_0^T}{\partial x_{P1}} 2H^T H \tag{B66}$$

$$\frac{\partial \alpha_2}{\partial \dot{x}_{P2}} = -\frac{\partial F D_0^T}{\partial x_{P2}} 2H^T H \tag{B67}$$

$$\frac{\partial \gamma}{\partial \dot{x}_{P1}} = \frac{\partial \dot{H}^T}{\partial \dot{x}_{P1}} \left(H\dot{x}_{P2} + C \right) + H^T \left(\frac{\partial \dot{H}}{\partial \dot{x}_{P1}} \dot{x}_{P2} + \frac{\partial \dot{C}}{\partial \dot{x}_{P1}} \right) \tag{B68}$$

$$\frac{\partial \gamma}{\partial \dot{x}_{P2}} = (\dot{H}^T H + H^T \dot{H}) + H^T \frac{\partial \dot{C}}{\partial \dot{x}_{P2}} \tag{B69}$$

$$\frac{\partial \Gamma}{\partial \dot{x}_{P2}} = I_{N\times N} \frac{\partial H}{\partial x_{P1}} H(\dot{x}_{P2} - F D_0) + (\dot{x}_{P2} - F D_0)^T \frac{\partial H}{\partial x_{P1}} H \tag{B70}$$

$\frac{\partial \gamma}{\partial \dot{x}_{P1}}$ (B68) and $\frac{\partial \gamma}{\partial \dot{x}_{P2}}$ (B69) can be simplified even further. Next, we highlight the following relations that will be used to simplify these terms. First, $\dot{H}$ can be computed using the chain rule in the following way:

$$\dot{H} = \frac{\partial H}{\partial x_{P1}} \frac{\partial x_{P1}}{\partial t} = \frac{\partial H}{\partial x_{P1}} \dot{x}_{P1} \tag{B71}$$

Second, $\dot{C}$ can also be computed similarly using the chain rule:

$$\dot{C} = \frac{\partial C}{\partial x_{P1}} \dot{x}_{P1} + \frac{\partial C}{\partial x_{P2}} \dot{x}_{P2} \tag{B72}$$

Lastly, recall that H is symmetric, therefore:

$$\frac{\partial H}{\partial x_{P1}} = \frac{\partial H^T}{\partial x_{P1}} \tag{B73}$$

Using these three relations we can obtain the following:

$$\frac{\partial \dot{H}}{\partial \dot{x}_{P1}} = \frac{\partial H}{\partial x_{P1}} \tag{B74}$$

$$\frac{\partial \dot{C}}{\partial \dot{x}_{P1}} = \frac{\partial C}{\partial x_{P1}} \tag{B75}$$

Using (B74) and (B75) to simplify $\frac{\partial \gamma}{\partial \dot{x}_{P1}}$ (B68) yields:

$$\frac{\partial \gamma}{\partial \dot{x}_{P1}} = \frac{\partial H}{\partial x_{P1}} \left(H\dot{x}_{P2} + C \right) + H \left(\frac{\partial H}{\partial x_{P1}} \dot{x}_{P2} + \frac{\partial C}{\partial x_{P1}} \right) \tag{B76}$$

we can apply similar line of reasoning to obtain a simplification for $\frac{\partial \gamma}{\partial \dot{x}_{P2}}$ (B69)

$$\frac{\partial \gamma}{\partial \dot{x}_{P2}} = \dot{H}H + H(\dot{H} + \frac{\partial C}{\partial x_{P2}}) \tag{B77}$$

Lastly, the derivative of Ω wrt $[D^2\xi]_i$ is given by

$$\frac{d\Omega}{d[D^2\xi]_i} = 2 \begin{bmatrix} I_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & (H^T H)^{-1} \frac{\partial \gamma}{\partial \ddot{x}_{P2}} \end{bmatrix} \tag{B78}$$

where

$$\frac{\partial \gamma}{\partial \ddot{x}_{P2}} = H^T H \tag{B79}$$

This simplifies to

$$\frac{d\Omega}{d[D^2\xi]_i} = 2\begin{bmatrix} I_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & I_{N\times N} \end{bmatrix} \tag{B80}$$

Observing the terms from Equations (B49) – (B60), we have that $\frac{d\Omega}{d\xi_i}$ (B44) is a function of the rigid body dynamics and its higher-order derivatives. Namely,

$$\frac{d\Omega}{d\xi_i}\left(H, \dot{H}, C, \dot{C}, FD_0, \frac{\partial H}{\partial x_{P1}}, \frac{\partial^2 H}{\partial x_{P1}^2}, \frac{\partial \dot{H}}{\partial x_{P1}}, \frac{\partial FD_0}{\partial x_{P1}}, \frac{\partial FD_0}{\partial x_{P2}}, \frac{\partial C}{\partial x_{P1}}, \frac{\partial C}{\partial x_{P2}}, \frac{\partial \dot{C}}{\partial x_{P1}}, \frac{\partial \dot{C}}{\partial x_{P2}}, \frac{\partial^2 FD_0}{\partial x_{P1}^2}, \frac{\partial^2 FD_0}{\partial x_{P2}^2}, \frac{\partial^2 FD_0}{\partial x_{P1}\partial x_{P2}}\right)$$

Similarly, observing the terms from Equations (B66), (B67), (B70), (B76) and (B77) , we have that $\frac{d\Omega}{d[D\xi]_i}$ (B61) is a function of the rigid body dynamics and its first-order derivatives. Specifically,

$$\frac{d\Omega}{d[D\xi]_i}\left(H, C, \dot{H}, FD_0, \frac{\partial H}{\partial x_{P1}}, \frac{\partial FD_0}{\partial x_{P1}}, \frac{\partial FD_0}{\partial x_{P2}}, \frac{\partial C}{\partial x_{P1}}, \frac{\partial C}{\partial x_{P2}}\right)$$

and from (B80), we have

$$\frac{d\Omega}{d[D^2\xi]_i} = 2\begin{bmatrix} I_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & I_{N\times N} \end{bmatrix}$$

■

C. Rapidly evaluating the AGHF (B5) *and its Jacobian* (B37)

This section presents several algorithms to rapidly evaluate the AGHF in (B5) and its Jacobian (B37) using spatial vector algebra based rigid body dynamics algorithms and discusses the computational efficiency of these algorithms.

1) Rapidly evaluating the AGHF (B5)

To efficiently evaluate the AGHF at each of the collocation nodes, one can use Theorem B3. Algorithm B1 shows how to leverage spatial vector algebra and some state-of-the-art dynamics algorithms [5] to rapidly compute the expressions introduced in Algorithm B1.

Algorithm B1 Leveraging Spatial Vector Algebra to Compute Ω (15)

Require: x, $\dot{x}$, $\ddot{x}$, k

H, $\dot{H} \leftarrow$ `CRBA_D`$(x_{P1}, \dot{x}_{P1})$

$C \leftarrow$ `RNEA`$(x_{P1}, x_{P2}, \mathbf{0})$

$\frac{\partial C}{\partial x_{P1}}$, $\frac{\partial C}{\partial x_{P2}} \leftarrow$ `RNEA_D`$(x_{P1}, x_{P2}, \mathbf{0})$

$\dot{C} \leftarrow \frac{\partial C}{\partial x_{P1}}\dot{x}_{P1} + \frac{\partial C}{\partial x_{P2}}\dot{x}_{P2}$

$\frac{\partial FD_0}{\partial x_{P1}}$, $\frac{\partial FD_0}{\partial x_{P2}}$, H^{-1}, $FD_0 \leftarrow$ `ABA_D`$(x_{P1}, x_{P2}, \mathbf{0})$

Compute Ω_1 (B6), Ω_2 (B7) and Ω_3 (B8)

model_gravity $\leftarrow 0$

$\frac{\partial H}{\partial x_{P1}}(\dot{x}_{P2} - FD_0) \leftarrow$ `RNEA_D`$(x_{P1}, \mathbf{0}, \dot{x}_{P2} - FD_0)$

$\omega_4 \leftarrow 2\frac{1}{k}\left[\frac{\partial H}{\partial x_{P1}}(\dot{x}_{P2} - FD_0)\right]^T H(\dot{x}_{P2} - FD_0)$

$\Omega_4 \leftarrow \begin{bmatrix}\omega_4^T & \mathbf{0}^T\end{bmatrix}^T$

Compute Ω (15)

By using recursive algorithms based on spatial vector algebra to compute the necessary dynamics terms in (15), this approach provides a substantial speedup over [6]. For example, algorithms like the Recursive Newton-Euler Algorithm (RNEA) achieve $\mathcal{O}(N)$ complexity for systems with N bodies, efficiently propagating forces and accelerations throughout the robot's kinematic chain and enabling faster, more scalable evaluation of the dynamics terms. This combination of recursive methods allows for a

more rapid evaluation of the AGHF's right-hand side compared to the original approach in [6], especially for higher dimensional systems. Table I in Section V of [1], shows how well this algorithm scales with increasing system dimension compared to [6].

In Algorithm B1, we begin by computing H and $\dot{H}$ using a modified version of the Composite Rigid Body Algorithm (`CRBA`) (Line 1). This modified version leverages the chain rule to compute the time derivatives of the various spatial quantities as we traverse the rigid body tree, yielding the time derivative of the mass matrix ($\dot{H}$). This modified version, we call `CRBA_D`, is a worst-case $\mathcal{O}(N^3)$ algorithm. We then use the Recursive Newton Euler Algorithm (`RNEA`) to rapidly compute C ($\mathcal{O}(N)$) (Line 2) and an extended version (`RNEA_D`) that computes `RNEA`'s derivatives with respect to q, $\dot{q}$ and $\ddot{q}$ to compute the derivatives of C with respect to to x_{P1} and x_{P2} ($\mathcal{O}(N^2)$ worst-case) (Line 3). These are all used to compute $\dot{C}$ (Line 4).

Next, we use the algorithm introduced in [7] to compute multiple partial derivatives of the Forward Dynamics when $u = \mathbf{0}$ (Line 5) leveraging the Articulated Body Algorithm (`ABA`). We denote this as `ABA_D`. This is a worst-case $\mathcal{O}(N^3)$ algorithm [8]. Utilizing the earlier results, we compute Ω_1, Ω_2 and Ω_3 (Line 6). Next, we compute $\frac{\partial H}{\partial x_{P1}}(\dot{x}_{P2} - FD_0)$ efficiently by setting the gravity term used by our dynamics model (model_gravity) to zero (Line 7) and using `RNEA_D` with zero velocity and setting the acceleration to $(\dot{x}_{P2} - FD_0)$, which avoids explicitly computing the tensor $\frac{\partial H}{\partial x_{P1}}$ (Line 8). Lastly, we perform a matrix-vector multiplication to compute ω_4 (Line 9) and stack the vector to obtain Ω_4 (Line 10). With all the terms computed we can compute Ω using (15) (Line 11).

Combining these operations with the $\mathcal{O}(N^3)$ matrix-matrix multiplications needed to compute (15), results in a worst-case $\mathcal{O}(N^3)$ algorithm for computing the AGHF right-hand side. Figure 4 shows the mean and standard deviation of the computation time of (15) as the number of bodies (N) of the system increases from 2 (Double Pendulum) to 22 (Digit). While the worst-case computational complexity is $\mathcal{O}(N^3)$, the results from Figure 4 indicate that the algorithm scales more efficiently in practice. Specifically, the polynomial line of best fit lacks a significant N^3 term, suggesting that the computational time scales approximately quadratically with the number of bodies N within the observed range.

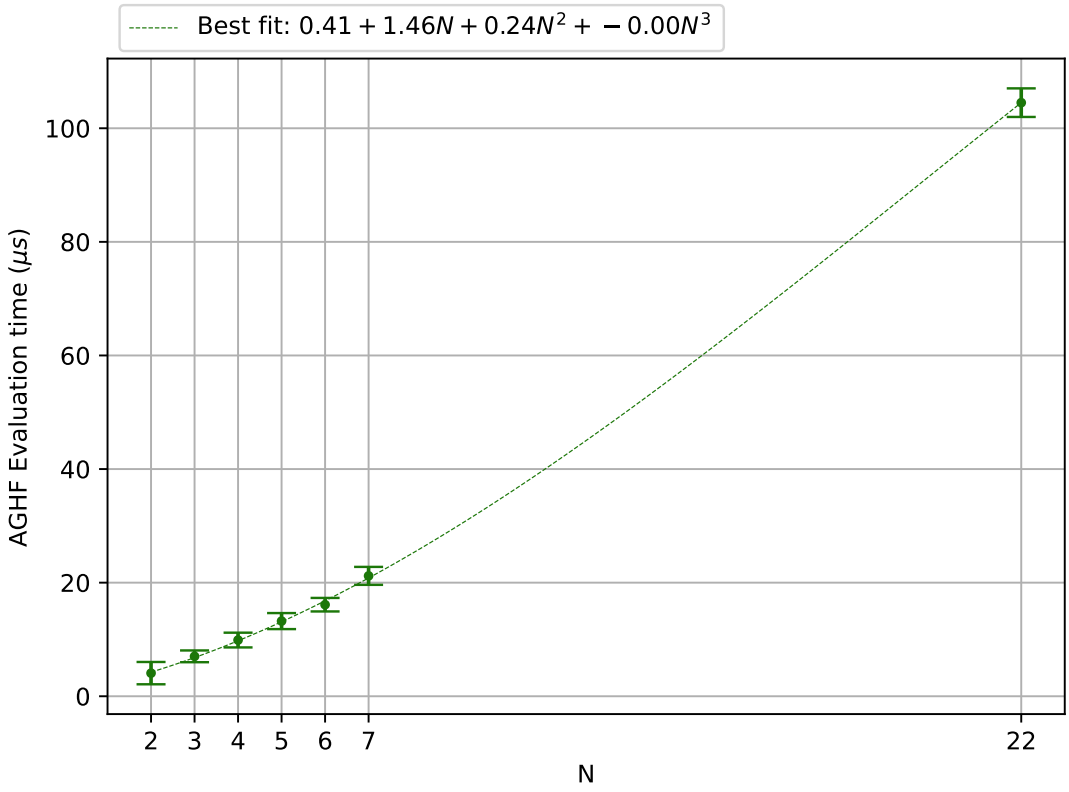

Fig. 4: Scaling trend of the mean evaluation times (in μs) of the right-hand side of the AGHF using Algorithm B1 as the number of bodies (N) increases from 2 to 22. The systems with N between 2 and 6 are penduli systems (i.e., 2-link pendulum, 3-link pendulum, etc.). The $N = 7$ system is the Kinova Gen3 and the $N = 22$ system is the pinned Digit V3 biped. Each robot's AGHF RHS was evaluated at a 1000 random robot configurations. Notice the polynomial line of best fit lacks a significant N^3 term, suggesting that the right-hand side computation time scales approximately quadratically with N in practice.

2) Rapidly Evaluating the AGHF Jacobian (B37)

To efficiently evaluate the AGHF Jacobian at each of the collocation nodes, one can use Theorem B4. Algorithm B2 shows how to leverage spatial vector algebra and state-of-the-art dynamics algorithms [5] to rapidly compute this Jacobian in (B37).

We begin by first computing H, $\dot{H}$, $\frac{\partial H}{\partial x_{P1}}$ and $\frac{\partial^2 H}{\partial x_{P1}^2}$ (Line 1) using a modified `CRBA` algorithm where we compute $\dot{H}$ and the first and second derivatives of H with respect to x_{P1} using the chain rule. This modified version, we call `CRBA_2D`, is a worst-case $\mathcal{O}(N^4)$ algorithm. Similar to Appendix B-C1, we use `ABA_D` (worst-case $\mathcal{O}(N^3)$[8]) to compute the partial derivatives of the Forward Dynamics (Line 2). We then use `RNEA` to compute C ($\mathcal{O}(N)$) (Line 3) and `RNEA_D` to compute the derivatives of C (Line 4) once more (($\mathcal{O}(N^2)$ worst-case). Next we use these derivatives to compute $\dot{C}$ (Line 5). Next, the function `get_Hdot_D` computes $\frac{\partial \dot{H}}{\partial x_{P1}}$ by applying the chain rule to $\frac{\partial^2 H}{\partial x_{P1}^2}$ and $\dot{x}_{P1}$ (Line 6). We then use `RNEA_2D`, which computes the second derivatives of the Inverse Dynamics (ID), with the acceleration passed in as zero to get the second derivatives of C (Line 7). Next, `RNEA_2D` with the acceleration set to FD_0 (which is the acceleration of the system, with $u = 0$) allows us to compute the derivatives of the inverse dynamics (Line 8). The inverse dynamics derivatives computed using the acceleration from FD_0 are needed to to rapidly compute the second derivatives of FD_0 using the algorithm proposed in [4], `ABA_2D` (Line 9). The function `get_Cdot_D`, similar to `get_Hdot_D`, computes the partial derivatives of $\dot{C}$ using the chain rule with the second derivatives of C and $\dot{x}_{P1}$ and $\dot{x}_{P2}$ (Line 10). With all these terms computed we then evaluate $J_{\Xi_i}(s)$ (B37) (Line 11). Overall, combining these operations with the $\mathcal{O}(N^4)$ tensor-matrix multiplications needed to compute (B37), results in an $\mathcal{O}(N^4)$ algorithm for computing $J_{\Xi_i}(s)$. Figure 5 shows the mean and standard deviation of the computation time of $J_{\Xi_i}(s)$ as the number of bodies (N) of the system increases from 2 (Double Pendulum) to 22 (Digit). While the worst-case computational complexity is $\mathcal{O}(N^4)$, the results from Figure 5 indicate that the algorithm scales more efficiently in practice. Specifically, the polynomial line of best fit lacks a significant N^4 term, suggesting that the Jacobian computational time scales approximately cubically with the number of bodies N within the observed range.

Algorithm B2 Leveraging Spatial Vector Algebra to Compute $J_{\Xi_i}(s)$ (B37)

Require: x, $\dot{x}$, $\ddot{x}$, k

1: H, $\dot{H}$, $\frac{\partial H}{\partial x_{P1}}$, $\frac{\partial^2 H}{\partial x_{P1}^2} \leftarrow$ `CRBA_2D`$(x_{P1}, \dot{x}_{P1})$

2: $\frac{\partial FD_0}{\partial x_{P1}}$, $\frac{\partial FD_0}{\partial x_{P2}}$, H^{-1}, $FD_0 \leftarrow$ `ABA_D`$(x_{P1}, x_{P2}, \mathbf{0})$

3: $C \leftarrow$ `RNEA`$(x_{P1}, x_{P2}, \mathbf{0})$

4: $\frac{\partial C}{\partial x_{P1}}$, $\frac{\partial C}{\partial x_{P2}} \leftarrow$ `RNEA_D`$(x_{P1}, x_{P2}, \mathbf{0})$

5: $\dot{C} \leftarrow \frac{\partial C}{\partial x_{P1}}\dot{x}_{P1} + \frac{\partial C}{\partial x_{P2}}\dot{x}_{P2}$

6: $\frac{\partial \dot{H}}{\partial x_{P1}} \leftarrow$ `get_Hdot_D`$(\dot{x}_{P1}, \frac{\partial^2 H}{\partial x_{P1}^2})$

7: $\frac{\partial^2 C}{\partial x_{P1}^2}$, $\frac{\partial^2 C}{\partial x_{P2}^2}$, $\frac{\partial^2 C}{\partial x_{P1}\partial x_{P2}} \leftarrow$ `RNEA_2D`$(x_{P1}, x_{P2}, \mathbf{0})$

8: $\frac{\partial^2 ID}{\partial x_{P1}^2}$, $\frac{\partial^2 ID}{\partial x_{P2}^2}$, $\frac{\partial^2 ID}{\partial x_{P1}\partial x_{P2}} \leftarrow$ `RNEA_2D`(x_{P1}, x_{P2}, FD_0)

9: $\frac{\partial^2 FD_0}{\partial x_{P1}^2}$, $\frac{\partial^2 FD_0}{\partial x_{P2}^2}$, $\frac{\partial^2 FD_0}{\partial x_{P1}\partial x_{P2}} \leftarrow$ `ABA_2D`$(\frac{\partial FD_0}{\partial x_{P1}}, \frac{\partial FD_0}{\partial x_{P2}}, H^{-1}, \frac{\partial H}{\partial x_{P1}}, \frac{\partial^2 ID}{\partial x_{P1}^2}, \frac{\partial^2 ID}{\partial x_{P2}^2}, \frac{\partial^2 ID}{\partial x_{P1}\partial x_{P2}})$

10: $\frac{\partial \dot{C}}{\partial x_{P1}}$, $\frac{\partial \dot{C}}{\partial x_{P2}} \leftarrow$ `get_Cdot_D`$(\dot{x}_{P1}, \dot{x}_{P2}, \frac{\partial^2 C}{\partial x_{P1}^2}, \frac{\partial^2 C}{\partial x_{P2}^2}, \frac{\partial^2 C}{\partial x_{P1}\partial x_{P2}}, \frac{\partial^2 C}{\partial x_{P2}\partial x_{P1}})$

11: **Compute** $J_{\Xi_i}(s)$ (B37)

D. Pseudospectral Method for solving the AGHF

The PHLAME algorithm is summarized in Algorithm B3. It first requires one to specify some initial curve x_0 with initial and terminal points as $\mathtt{x}_0$ and $\mathtt{x}_f$ respectively. Along with the initial curve one must specify the number of pseudospectral nodes p, the penalty term k to be used in G, and the final s in the domain of the homotopy s_{max}. Algorithm B3 then sets the values of the initial pseudospectral nodes, $\xi(0)$, equal to the initial curve (Line 1). Then an ODE solver (e.g., Runge Kutta or Adams Bashforth Method [9]) can be used to simulate solution of the AGHF PDE at each of the collocation nodes (Line 2).

Algorithm B3 PHLAME

Require: $x_0 : [0, T] \to \mathbb{R}^{2N}$ s.t. $x_0(-1) = \mathtt{x}_0$, $x_0(1) = \mathtt{x}_f$, $p \in \mathbb{N}$, k and s_{max}.

1: $\xi_i^T(0) \leftarrow x_0(t_i)$ for $i \in \{1, \ldots, p-1\}$ (20).

2: **Compute** $\xi(s_{max})$ using an ODE Solver.

3: **Compute** $u(t_i)$ using $\xi(s_{max})$ and (13).

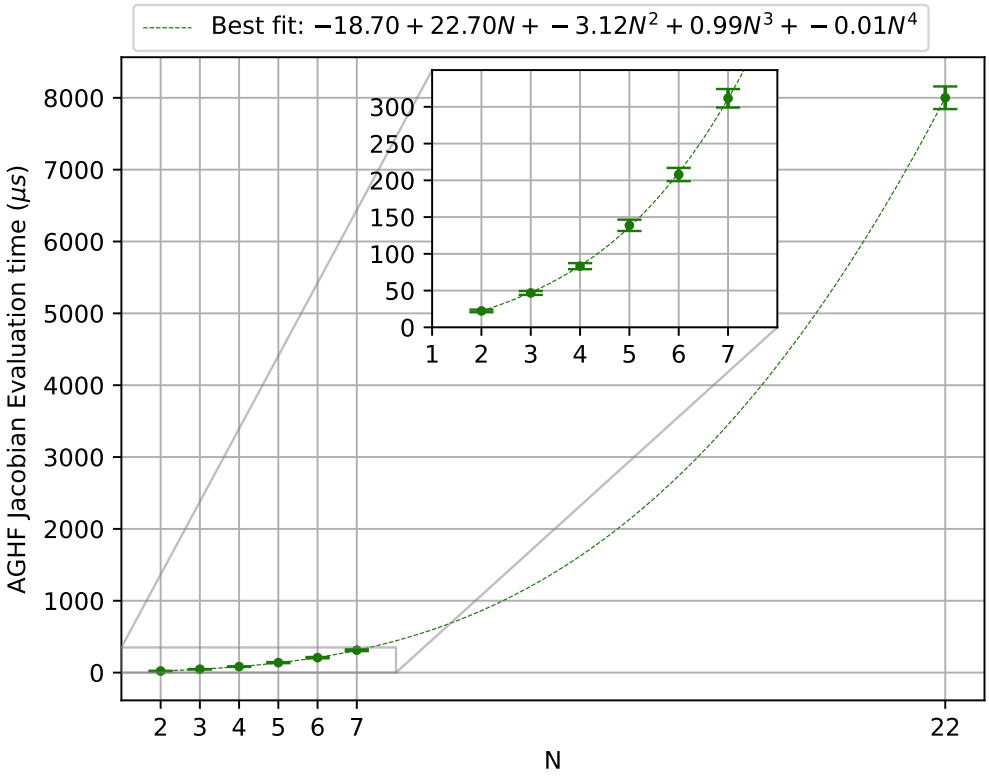

Fig. 5: Scaling trend of the mean evaluation times (in μs) of the AGHF Jacobian using Algorithm B2 as the number of bodies (N) increases from 2 to 22. The systems with N between 2 and 6 are penduli systems (i.e., 2-link pendulum, 3-link pendulum, etc.). The $N = 7$ system is the Kinova Gen3 and the $N = 22$ system is the pinned Digit V3 biped. Each robot's AGHF Jacobian was evaluated at a 1000 random robot configurations. Notice the polynomial line of best fit lacks a significant N^4 term, suggesting that the Jacobian computation time scales approximately cubically with N in practice.

APPENDIX C
RESULTS APPENDIX

This sections provides additional details on the experiments and results presented in [1].

A. *Solver Setup*

All experiments in [1] were run on a Ubuntu 22.04 machine with an Intel Xeon Platinum 8170M @ 208x 3.7GHz CPU. Each of the evaluated numerical methods requires an initial and final desired state and an initial guess for the initial trajectory x_0 along with the selection of solver specific parameters. Note the trajectory optimization methods also require an initial control input $u_0 : [0, T] \to \mathbb{R}^m$. Throughout this section, the term experiment refers to a tuple of experimental parameters consisting of the robotic platform, $\mathtt{x}_0$, $\mathtt{x}_f$, and a set of obstacles (if present).

To perform a fair comparison, we ran a grid search over the parameter space to obtain the best possible solver parameters. Each grid search was ran in parallel with a timeout per parameter set. The timeout for the systems with ≤ 5 DOF was 3 minutes and for the rest, 5 minutes. Then, we re-ran the experiments sequentially with the best solver specific parameters per method to obtain timings. The best solver specific parameter is defined as the one that yields the lowest solve time while producing a success for the experiment as defined above.

We next describe the parameters that make up the grid search for each method. Note that these parameters and how they are varied are summarized in Apppendix C-B.

As part of the grid search for Crocoddyl and Aligator, we considered different types of initial guesses to pass to the optimizer based on the examples given in their publicly available code. For Aligator, we considered the following initial guesses: (1) "Zeros," which corresponds to an initial guess that is zero for all time for x_0 and u_0; (2) "Line and RNEA," which corresponds to an initial guess where x_0 is a line connecting $\mathtt{x}_0$ and $\mathtt{x}_f$ and u_0 is equal to applying RNEA using x_0 and $\dot{x}_0$ (here, $\dot{x}_0$ is obtained by fitting a chebyshev polynomial to x_0 and taking it's derivative); (3) "Rollout and Zero," which corresponds to an initial guess where u_0 is zero and x_0 is the result of forward simulating the dynamical system using zero input; and (4) "Rollout and Constant," which corresponds to an initial guess for u_0 that is constant and is equal to applying RNEA to the initial $\mathtt{x}_0$ while assuming that $\ddot{q}(0) = 0$ and x_0 is equal to forward simulating the system using that control input. For Crocoddyl, we

considered the following initial guesses: "Zeros" and "Line and RNEA," as defined in the Aligator case, and "Constants," which corresponds to setting $x(t) = \mathtt{x}_0$ for all t and u_0 equal to a constant that is equal to applying RNEA to the initial $\mathtt{x}_0$ while assuming that $\ddot{q}(0) = 0$ and x_0. For PHLAME, we only considered the initial guess "Line', which corresponds to an initial guess where x_0 is a line connecting $\mathtt{x}_0$ and $\mathtt{x}_f$. Note that for our method we do not need to specify an initial guess u_0.

Aligator and Croccodyl have several parameters that are specific to their implementation. First, each method relies upon discretizing time and allow a user to specify a time discretization, δ_t. Second, each method allows one to include a running cost. We choose this running cost to be the 2-norm of the input added to the 2-norm of the state with weights w_u and w_x, respectively. Third, each method allows one to include a terminal cost with weight w_{xf}. We choose this to be the 2-norm of the difference of the final state from $\mathtt{x}_f$. For Aligator, we also consider the parameters μ_{init} and ϵ_{tol}, where the former corresponds to the initial value of the augmented Lagrangian penalty parameter and the latter to the solver tolerance. Lastly, for Aligator we also add an equality constraint that enforces the desired final state $\mathtt{x}_f$ and inequality constraints that enforce that the joint frames are not inside any of the sphere obstacles.

As for PHLAME. First, p is the degree of the polynomial that represents the solution. Second, k is a penalty that ensures dynamic feasibility and was first introduced in (11). Third, s_{max} corresponds to the maximum "time" that the PDE has to evolve. Fourth, only for the experiments that have obstacles (inequality constraints) we also consider the parameters k_{cons}, introduced in (A2) and c_{cons}, introduced in (A3) which control the weight of the constraint satisfaction and the sharpness of the activation function respectively.

In summary, for Crocoddyl in each experiment we perform a grid search of Initial guess, δ_t, w_u, w_x, and w_{xf}. For Aligator we search over the Initial guess, $\delta_t, w_u, w_x, w_{xf}, \mu_{init}$ and ϵ_{tol}. For unconstrained PHLAME, p, s_{max} and k and for constrained PHLAME p, s_{max}, k, c_{cons} and k_{cons}.

B. Grid parameters for each Solver

This section contains all the varied parameters that we grid search over and use for the experiments in [1, Section V]:

Parameter	**Grid values**
w_u	$[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$
w_x	$[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.0, 1, 10]$
w_{xf}	$[10^{-4}, 1.0, 10, 1000]$
δ_t	$[10^{-2}, 10^{-3}, 10^{-4}]$
Initial guess	[Zeros, Line and RNEA, Constant]

TABLE VII: Parameter Values for Crocoddyl unconstrained

Parameter	**Grid values**
w_u	$[10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$
w_x	$[10^{-6}, 10^{-4}, 10^{-3}, 0.0, 1, 100, 1000]$
w_{xf}	$[10^{-6}, 10^{-4}, 1.0, 10, 1000]$
δ_t	$[10^{-2}, 10^{-3}]$
ϵ_{tol}	$[10^{-3}, 10^{-4}, 10^{-7}]$
μ_{init}	$[10^{-2}, 10^{-7}, 10^{-8}]$
Initial guess	[Zeros, Line and RNEA, Rollout and Zero, Rollout and Constant]

TABLE VIII: Parameter Values for Aligator with obstacles

Parameter	**Grid values**
s_{max}	$[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 5, 10, 25, 50, 100]$
k	$[10^3, 10^4, 10^5, 10^6, 10^7, 10^8, 10^9, 10^{10}]$
p	$[5, 6, 7, 8, 9, 10, 15]$
Initial guess	[Line]

TABLE IX: Parameter Values for PHLAME for pendulum swingup

Parameter	**Grid values**
s_{max}	$[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 100]$
k	$[10^4, 10^5, 10^6, 10^7, 10^8, 10^9, 10^{10}]$
p	$[5, 6, 7, 8, 9, 10, 15]$
Initial guess	[Line]

TABLE X: Parameter Values for PHLAME for Kinova without obstacles

Parameter	Grid values
s_{max}	$[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 100]$
k	$[10^3, 10^4, 10^5, 10^6, 10^7, 10^8, 10^9, 10^{10}]$
p	$[5, 6, 7, 8, 9, 10]$
Initial guess	[Line]

TABLE XI: Parameter Values for PHLAME for Digit without obstacles

Parameter	Grid values
s_{max}	$[10^{-4}, 10^{-2}, 10^{-1}, 1, 5, 10, 25, 50, 100]$
k	$[10^5, 10^6, 10^7, 10^8, 10^9, 10^{10}]$
p	$[5, 6, 7, 8, 9, 10, 15]$
c_{cons}	$[1, 50, 200]$
k_{cons}	$[10^5, 10^6, 10^7, 10^8, 10^9, 10^{10}]$
Initial guess	[Line]

TABLE XII: Parameter Values for PHLAME Kinova with Constraints

Parameter	Grid values
s_{max}	$[10^{-4}, 10^{-2}, 10^{-1}, 1, 5, 10, 25, 50, 100]$
k	$[10^5, 10^6, 10^7, 10^8, 10^9, 10^{10}]$
p	$[5, 6, 7, 8, 9, 10, 15]$
c_{cons}	$[1, 50, 200]$
k_{cons}	$[10^5, 10^6, 10^7, 10^8, 10^9, 10^{10}]$
Initial guess	[Line]

TABLE XIII: Parameter Values for PHLAME for Digit with Constraints

C. Best Parameters for the Experiments

This section provides details on the best parameter set for each evaluated method in the experiments in [1, Section V].

1) Trajectory Optimization: Kinova Arm without Obstacles

The best parameter set for PHLAME was $s_{max} = 0.1, k = 10^9, p = 9$. The best parameter set for Crocoddyl was $w_x = 0.0001, w_u = 0.0001, \delta t = 0.0001, w_{xf} = 1$, and Initial Condition = "Zeros".

2) Trajectory Optimization: Digit without Obstacles

The best parameter set for PHLAME was $s_{max} = 1, k = 10^7, p = 6$. The best parameter set for Crocoddyl was $w_x = 0.001, w_u = 0.0001, \delta t = 0.001, w_{xf} = 1000$, and Initial Condition = "Rollout and Constant".

3) Trajectory Optimization: Kinova Arm with Obstacles

The best parameter set for PHLAME was $s_{max} = 0.1, k = 10^9, p = 8, k_{cons} = 10^9, c_{cons} = 1$. The best parameter set for Aligator was $w_x = 0.001, w_u = 0.0001, \delta t = 0.001, w_{xf} = 1$, and Initial Condition = "Line and RNEA", $\epsilon_{tol} = 0.001, \mu_{init} = 10^{-7}$.

4) Trajectory Optimization: Digit with obstacles

For a error threshold of $\epsilon = 0.05$ the best parameter set for PHLAME was $s_{max} = 1.0, k = 10^7, p = 6, k_{cons} = 10^5, c_{cons} = 200$. For Aligator none of the parameter sets yielded a success when the allowable final state error $\epsilon = 0.05$ as with all the other experiments. With a larger error threshold (ϵ = 0.25), the best parameter set for Aligator was $w_x = 1.0, w_u = 0.01, \delta t = 0.01, w_{xf} = 10^{-6}$, and Initial Condition = "Zeros", $\epsilon_{tol} = 0.001, \mu_{init} = 10^{-8}$.

REFERENCES

[1] C. Enninful Adu, C. E. Ramos Chuquiure, B. Zhang, and R. Vasudevan, *Bring the heat: Rapid trajectory optimization with pseudospectral techniques and the affine geometric heat flow equation*, 2024.

[2] Y. Fan, "Robot motion planning via curve shortening flows," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2021.

[3] L. Pareschi, G. Russo, *et al.*, "Implicit-explicit runge-kutta schemes for stiff systems of differential equations," *Recent trends in numerical analysis*, vol. 3, pp. 269–289, 2000.

[4] S. Singh, R. P. Russell, and P. M. Wensing, "On second-order derivatives of rigid-body dynamics: Theory & implementation," *IEEE Transactions on Robotics*, 2024.

[5] J. Carpentier, G. Saurel, G. Buondonno, *et al.*, "The pinocchio c++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, 2019, pp. 614–619.

[6] S. Liu, Y. Fan, and M.-A. Belabbas, "Affine geometric heat flow and motion planning for dynamic systems," *IFAC-PapersOnLine*, vol. 52, no. 16, pp. 168–173, 2019, 11th IFAC Symposium on Nonlinear Control Systems NOLCOS 2019.

[7] J. Carpentier and N. Mansard, "Analytical Derivatives of Rigid Body Dynamics Algorithms," in *Robotics: Science and Systems (RSS 2018)*, Pittsburgh, United States, Jun. 2018.

[8] S. Singh, R. P. Russell, and P. M. Wensing, "Efficient analytical derivatives of rigid-body dynamics using spatial vector algebra," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, 1776–1783, Apr. 2022.

[9] J. P. Boyd, *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.