Supplementary Material
Learning Forward Looking Adaptation to Dynamic Payloads for Quadruped Locomotion via Physics-Informed Neural Networks

I. Whole-Body Impulse Control Details

This section provides details on the Whole-Body Impulse Control (WBIC) [1] controller used as $\Phi(\cdot)$ in FLAP. This controller considers the following general equations of motion:

$$\mathbf{A}\ddot{\boldsymbol{\rho}} + \mathbf{b} + \mathbf{g} = \begin{bmatrix} \mathbf{0}^6 \\ \boldsymbol{\tau}^{\mathbf{ex}} \end{bmatrix} + \mathbf{J}_c^\top \mathbf{f}^r \tag{1}$$

where $\ddot{\boldsymbol{\rho}} = [\ddot{\mathbf{p}}, \ddot{\Theta}, \ddot{\mathbf{q}}]^\top$ and $\mathbf{A}$, $\mathbf{b}$, $\mathbf{g}$, $\mathbf{J}_c^\top$, and $\mathbf{f}^r$ are the generalized mass matrix, Corilois force vector, gravitation force vector, contact Jacobian, and ground reaction force vector respectively. Additionally $\boldsymbol{\tau}^{\mathbf{ex}}$ are the expected leg joint torques and $\mathbf{0}^6$ represents a six dimensional vector of zeros, corresponding to the six not directly controlled degrees of freedom of the torso.

The WBIC controller in [1] leverages the above definition of the whole body dynamics to track an optimal ground reaction force profile along side acceleration commands. This accomplished by first using the current state of the robot and the expected torso locomotion commands $\mathbf{b}^{ex}$ (defined as expected linear and angular velocities as in the main text) to find ground reaction forces that allow the robot to follow the desired trajectory. The optimal ground reaction force profile is computed using a real-time sequential quadratic programming (SQP)-based Nonlinear Model Predictive Control (NMPC) controller from [2]. The ground force reaction profile generated by the NMPC, denoted as $\mathbf{f}^r_{\text{MPC}}$, is then used to compute expected joint position $\mathbf{q}^{ex}$, velocity $\dot{\mathbf{q}}^{ex}$, acceleration $\ddot{\mathbf{q}}^{ex}$ and torque $\boldsymbol{\tau}^{ex}$ commands.

The WBIC computes the expected $\ddot{\mathbf{q}}^{ex}$, $\dot{\mathbf{q}}^{ex}$, and $\mathbf{q}^{ex}$ using an hierarchical inverse kinematics algorithm that uses null space projection to enforce a strict task priority over three control objectives: torso orientation, torso position, and foot position, in that order. The quadratic program in Eq. (2) is then used to refine $\mathbf{f}^r_{\text{MPC}}$ and $\ddot{\mathbf{q}}^{ex}$ using the whole body dynamics in Eq. (1).

$$\begin{aligned} \min_{\delta_{\mathbf{f}^r}, \delta_f} \quad & \delta_{\mathbf{f}^r}^\top Q \delta_{\mathbf{f}^r}^\top + \delta_f^\top Q \delta_f^\top \\ s.t. \quad & \\ & \mathbf{S}^f \left(\mathbf{A}\ddot{\boldsymbol{\rho}} + \mathbf{b} + \mathbf{g}\right) = \mathbf{S}^j \mathbf{J}_c^\top \mathbf{f}^r \\ & \ddot{\boldsymbol{\rho}} = \begin{bmatrix} \ddot{\mathbf{p}}^{ex} \\ \ddot{\boldsymbol{\Theta}}^{ex} \\ \ddot{\mathbf{q}}^{ex} \end{bmatrix} + \begin{bmatrix} \delta_f \\ \mathbf{0}^{n_j} \end{bmatrix} \\ & \mathbf{f}^r = \mathbf{f}^r_{\text{MPC}} + \delta_{\mathbf{f}^r} \\ & W\mathbf{f}^r \geq \mathbf{0} \qquad \text{(contact force constraint)} \end{aligned} \tag{2}$$

In Eq. (2), $\delta_{\mathbf{f}^r}$ and δ_f are slack variables used to adjust the ground reaction forces and torso's 6DOF acceleration respectively. Additionally, W denotes the contact constraint matrix and n_j the number of leg joints. Finally, using the results from Eq. (2), the WBIC computes the expected joint torques via:

$$\begin{bmatrix} - \\ \boldsymbol{\tau}^{ex} \end{bmatrix} = \mathbf{A}\ddot{\boldsymbol{\rho}} + \mathbf{b} + \mathbf{g} - \mathbf{J}_c^\top \mathbf{f}^r \tag{3}$$

II. Data Collection

A. Data Collection Settings

Training data was collected in the MuJoCo simulator with the robot being controlled by $\Phi(\cdot)$. During data collection a force equivalent to a 0, 1.0, 2.5, 5.0, 7.5, and 10.0 kg payload is applied above the geometric center of the robot's torso in the negative z-direction. Additionally, random torso velocity commands $\mathbf{b}^{ex}$ are sampled from the ranges defined in Table. 1 of the main text. For each payload condition, four data collection trail each lasting four minutes where performed, resulting in a total of 96 minutes worth of training data. Sampled at a rate of 500 Hz, this results in a total of 2.88 million time-steps. The forces and torques acting on the robot (including the torso and the feet contacts) are directly available from the MuJoCo simulator and are collected as a part of the training data. The ground reaction forces are not saved directly, but instead are first converted to joint torques with $\mathbf{J}_c^\top$ (also available through the MuJoCo simulator) which are then saved for use in offline training.

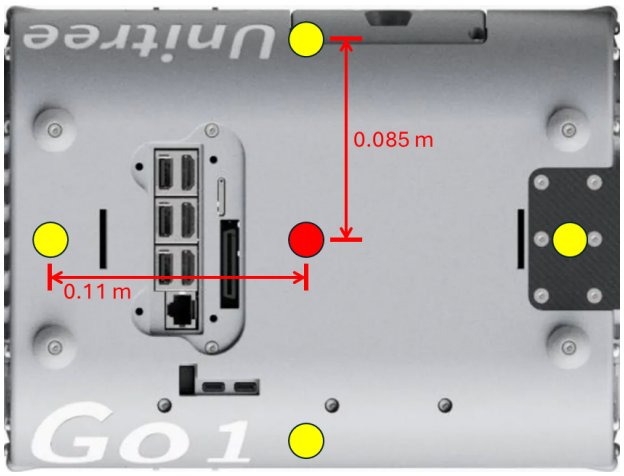

Fig. 7. Torso torques sampling positions. The red dot is above the geometric center of the quadruped's torso and is the position where the payload force is applied in the negative z-direction. The yellow dot's correspond to the sample locations for approximating dynamic payloads. Note the horizontal and vertical offsets from the red dot are symmetric about the x and y axis of the torso.

B. Approximating a Dynamic Payload

When a payload force is applied to the torso of the robot using the appropriate tools in MuJoCo, the dynamics of the robot are used to calculate the torques imparted on the torso based on the payload's mass and location (position where the force is applied to the torso). However, this is only done once, when the payload is applied, and not recalculated as the simulation progresses. Therefore, to simulated a dynamically changing payload, we additionally sampled torques at a rate of 10 Hz that are applied to the robot's torso in addition to the vertical force from the payload. For each payload mass used during data collection, we first recorded the torso torques generated by MuJoCo at five key positions: directly above the geometric center of the robot, and at the front, back, left, and right edges of the torso (Fig. 7). The mean and standard deviation of these torque values were used to parameterize a normal distribution, which was sampled throughout the simulation. This approach effectively approximates the impact of the COM of the payload shifting randomly throughout deployment with respect to the geometric center of the robot's torso.

III. Training

A. Algorithm for Constructing PINN Loss

Algorithm 2: Construct PINN Loss

Input: $\hat{\Delta}t_{t+1}, \hat{\mathbf{q}}_{t+1}, \hat{\mathbf{p}}^z_{t+1}, \hat{\Theta}_{t+1}, \hat{\dot{\mathbf{p}}}_{t+1}, \hat{\dot{\Theta}}_{t+1}, \hat{\boldsymbol{\tau}}^{net}_{t+1}, \mathbf{p}^{x,y}_{t+1}, \dot{\mathbf{q}}_{t+1}$

Output: $\mathcal{L}_{wb}, \mathcal{L}_{\dot{\mathbf{q}}_{t+1}}$

1 $\dot{\mathbf{q}}^{ad}_{t+1} \leftarrow f_{ad}(\hat{\Delta}t_{t+1}, \hat{\mathbf{q}}_{t+1})$, $\ddot{\mathbf{q}}^{ad}_{t+1} \leftarrow f_{ad}(\hat{\Delta}t_{t+1}, \dot{\mathbf{q}}^{ad}_{t+1})$

2 $\dot{\mathbf{p}}^{ad}_{t+1} \leftarrow f_{ad}(\hat{\Delta}t_{t+1}, \hat{\dot{\mathbf{p}}}_{t+1})$, $\dot{\Theta}^{ad}_{t+1} \leftarrow f_{ad}(\hat{\Delta}t_{t+1}, \hat{\dot{\Theta}}_{t+1})$

3 $\ddot{\boldsymbol{\rho}}^{p}_{t+1} \leftarrow \left[\dot{\mathbf{p}}^{ad}_{t+1}, \dot{\Theta}^{ad}_{t+1}, \ddot{\mathbf{q}}^{ad}_{t+1}\right]^\top$, $\dot{\boldsymbol{\rho}}^{p}_{t+1} \leftarrow \left[\hat{\dot{\mathbf{p}}}_{t+1}, \hat{\dot{\Theta}}_{t+1}, \dot{\mathbf{q}}^{ad}_{t+1}\right]^\top$, $\boldsymbol{\rho}^{p}_{t+1} \leftarrow \left[\mathbf{p}^{x,y}_{t+1}, \hat{\mathbf{p}}^{z}_{t+1}, \hat{\Theta}_{t+1}, \hat{\mathbf{q}}_{t+1}\right]^\top$

4 $\mathbf{A} \leftarrow$ pinocchio.RNEA($\boldsymbol{\rho}^{p}_{t+1}, \dot{\boldsymbol{\rho}}^{p}_{t+1}, \mathbf{0}$)

5 $\mathbf{b}, \mathbf{g} \leftarrow$ pinocchio.CRBA($\boldsymbol{\rho}^{p}_{t+1}$)

6 $\mathcal{L}_{\dot{\mathbf{q}}_{t+1}} \leftarrow \|\dot{\mathbf{q}}_{t+1} - \dot{\mathbf{q}}^{ad}_{t+1}\|^2$

7 $\mathcal{L}_{wb} \leftarrow \|\mathbf{A}\ddot{\boldsymbol{\rho}}^{p}_{t+1} + b + g - \hat{\boldsymbol{\tau}}^{net}_{t+1}\|^2$

8 **return** $\mathcal{L}_{\dot{\mathbf{q}}_{t+1}}, \mathcal{L}_{wb}$

The algorithm used to construct the PINN losses introduced in the main text is presented in Alg. 2. To calculate the terms $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{g}$ we utilize the Pinocchio rigid body dynamics python library [3]. Pinocchio's implementation of the recursive Newton-Euler algorithm (RNEA) is used to calculate the generalized mass matrix $\mathbf{A}$ and the composite rigid body algorithm (CRBA) calculates the Coriolis and gravity forces (returned as a single vector). Note, the Pinocchio library operates on is not designed for batch operations, so the steps taken in Alg. 2 must be performed for each sample in a batch. Specifically, we perform lines 1-5 for each sample and aggregate the results. They are then reshaped back into batches matching the original batch size and the loss terms are calculated in a batch-wise manner.

Algorithm 3: FLAP Training

Input: $\mathbf{D}$
Output: Trained model $\Psi(\cdot)$
1 Initialize $\Psi(\cdot)$
2 **while** ***not converged*** **do**
3 Sample $\mathbf{X}^p, \mathbf{x}^p, \Delta t_{t+1}, \mathbf{p}_{t+1}, \mathbf{\Theta}_{t+1}, \mathbf{q}_{t+1}, \dot{\mathbf{p}}_{t+1}, \dot{\mathbf{\Theta}}_{t+1}, \dot{\mathbf{q}}_{t+1}, \boldsymbol{\tau}^{net}_{t+1}$ from $\mathbf{D}$
4 $\mathbf{z}, \hat{\Delta} t_{t+1} \leftarrow \Psi_{ENC}(\mathbf{X}^p)$
5 $\hat{\mathbf{q}}_{t+1}, \hat{\boldsymbol{\tau}}^{net}_{t+1}, \hat{\mathbf{p}}^z_{t+1}, \hat{\mathbf{\Theta}}_{t+1}, \hat{\dot{\mathbf{p}}}_{t+1}, \hat{\dot{\mathbf{\Theta}}}_{t+1} \leftarrow \Psi_{FD}(\mathbf{z}, \hat{\Delta} t_{t+1}, \mathbf{x})$
6 $\hat{\mathbf{X}}^p \leftarrow \Psi_{DEC}(\mathbf{z})$
7 $\mathcal{L}_{\dot{\mathbf{q}}_{t+1}}, \mathcal{L}_{wb} \leftarrow PINN(\hat{\Delta} t_{t+1}, \hat{\mathbf{q}}_{t+1}, \hat{\mathbf{p}}^z_{t+1}, \hat{\mathbf{\Theta}}_{t+1}, \hat{\dot{\mathbf{p}}}_{t+1}, \hat{\dot{\mathbf{\Theta}}}_{t+1}, \hat{\boldsymbol{\tau}}^{net}_{t+1}, \mathbf{p}^{x,y}_{t+1}, \dot{\mathbf{q}}_{t+1})$
8 $\mathbf{s}^p_{t+1} \leftarrow [\mathbf{p}^z_{t+1}, \mathbf{\Theta}_{t+1}, \dot{\mathbf{p}}_{t+1}, \dot{\mathbf{\Theta}}_{t+1}]^\top$, $\hat{\mathbf{s}}^p_{t+1} \leftarrow [\hat{\mathbf{p}}^z_{t+1}, \hat{\mathbf{\Theta}}_{t+1}, \hat{\dot{\mathbf{p}}}_{t+1}, \hat{\dot{\mathbf{\Theta}}}_{t+1}]^\top$
9 $\mathcal{L}^p_n = \|\mathbf{s}^p_{t+1} - \hat{\mathbf{s}}^p_{t+1}\|^2 + \|\boldsymbol{\tau}^{net}_{t+1} - \hat{\boldsymbol{\tau}}^{net}_{t+1}\|^2$
10 $\mathcal{L}_{CE} \leftarrow \|\mathbf{X}^p - \hat{\mathbf{X}}^p\|^2 + (\Delta t_{t+1} - \hat{\Delta} t_{t+1})^2 + D_{KL}(\mathbf{z}|\mathcal{N}(0,1))$
11 Update $\Psi(\cdot)$ via PCGrad($\mathcal{L}^p_n, \mathcal{L}_{CE}, \mathcal{L}_{\dot{\mathbf{q}}_{t+1}}, \mathcal{L}_{wb}$)
12 **return** $\Psi(\cdot)$

B. Training Algorithm

We present our training procedure in Algorithm 3. Given a dataset $\mathbf{D}$, each epoch randomly samples a batch of training data on Line 3. The historical context $\mathbf{X}^p$ is used by the context encoder model $\Psi_{ENC}(\cdot)$ on to generate the context embedding $\mathbf{z}$ and predict the next time-step $\hat{\Delta} t_{t+1}$ line 4. Next, on line 5, $\mathbf{z}$ and $\hat{\Delta} t_{t+1}$ and the current time-steps input $\mathbf{x}^p$ are fed to the forward dynamics model $\Psi_{FD}(\cdot)$ to predict the next time-steps state $\mathbf{s}^p_{t+1}$. The decoder model $\Psi_{DEC}(\cdot)$ also uses $\mathbf{z}$ to reconstruct the historical context on line 6. For models trained using the physics informed whole body dynamics loss described in the main text, Alg. 2 is utilized to calculate the whole body dynamics and joint velocity PINN losses on line 7. On lines 9 and 10 the $\Psi_{FD}(\cdot)$'s next state prediction and the $\Psi_{ENC/DEC}(\cdot)$ variational auto encoding losses are calculated. Finally, the parameters of $\Psi(\cdot)$ are updated on line 11. Concretely, $\Psi(\cdot)$ is updated through the use of the *projecting conflicting gradients* (PCGrad) update mechanism introduced in [4]. This was done because it was found that the multi-task learning landscape introduced through the use of the PINN loss terms would result in the training process "overfitting" to one objective, improving it throughout training while the other terms worsened. Introducing PCGrad as a multi-task objective function resulted in much more stable training. To further improve training stability and generalization to real-world deployment conditions, the Lipschitz property of the $\Psi(\cdot)$ is regulated through layer-wise spectral normalization [5]–[7]. This is performed by calculating the L2 norm on the parameters of each layer in $\Psi(\cdot)$ and normalizing if the norm exceeds a threshold $\kappa > 0$. Spectral normalization is performed after every model update (line 11).

C. Training Parameters

TABLE III
TRAINING HYPERPARAMETERS

Symbol	Description	Value
-	Learning Rate	0.0001
-	Exponential LR Decay Rate	0.99
-	Number of Epochs	400
-	LR Decay Warm-up Epochs	100
-	Training Batch Size	4096
K	Context Encoder History Size	5
κ	Maximum Layer-wise Spectral Norm. of $\Psi(\cdot)$	6.0

Table III lists the relevant hyperparameters used throughout training. Both learning-based approaches explored in the main text (FLAP and F-NP) shared the same hyperparameters throughout training. An exponential learning rate scheduler was employed to gradually decrease the learning rate after an initial warm-up period during which the initial learning rate was held constant.

IV. IMPLEMENTATION

FLAP is implemented as a set of feedforward MLP models. The context encoder $\Psi_{ENC}(\cdot)$ consisted of three shared layers of size [128, 64, 18] followed by two distinct heads. One was of size 16 and generated the historical context encoding $\mathbf{z}$ while the other predicted the scalar value $\hat{\Delta} t_{t+1}$. The decoder $\Psi_{DEC}(\cdot)$ consisted of three layers of size [64, 128, 200] with the final output being of size 230. The forward dynamics model $\Psi_{FD}(\cdot)$ was an MLP with three shared layers of size [128, 64, 32] followed by four distinct prediction heads. The first predicted the next joint state $\hat{\mathbf{q}}_{t+1}$, the next predicted the

height and orientation of the torso $[\hat{\mathbf{p}}^z_{t+1}, \hat{\Theta}_{t+1}]$, the third predicting the torso's next velocity state $[\hat{\dot{\mathbf{p}}}_{t+1}, \hat{\dot{\Theta}}_{t+1}]$, and the last predicting the next time-steps net-forces/torques $\hat{\boldsymbol{\tau}}^{net}_{t+1}$. During training, all of the prediction heads of $\Psi_{FD}(\cdot)$ are used and updated. After training, the learned parameters of $\Psi_{ENC}(\cdot)$ and $\Psi_{FD}(\cdot)$ are compiled using torch script but exclude the parameters for all the prediction heads of $\Psi_{FD}(\cdot)$ other than those used to predict $\hat{\mathbf{q}}_{t+1}$. Not including the parameters for the prediction heads that predict the values used to construct the PINN loss in the deployment model reduce the computational demands of the model, enabling high-speed inference of 0.5 kHz.

V. Simulated Evaluation

Simulated evaluation of FLAP and the baseline approaches consisted of performing different locomotion behaviors while transporting dynamic payloads. The locomotion behaviors where consistent between all evaluated payloads. During evaluation, a locomotion behavior was defined as a set of expected torso velocities $\mathbf{b}^{ex}$ and a command duration. For example, the locomotion behavior $[0.8, 0, 0, 15]$ would represents a command of a 0.8 m/s linear velocity in the x-direction of the robot torso's frame with no lateral or yaw velocity for a duration of 15 seconds. following this notation, the four locomotion behaviors evaluated in these experiments are defined as:

1) Forwards/Backwards Motions: $[[0.5, 0, 0, 8], [1.0, 0, 0, 5], [-0.5, 0, 0, 5], [-1.0, 0, 0, 5]]$
2) Turning In-Place: $[[0, 0, 1.0, 8], [0, 0, 2.0, 5], [0, 0, -1.0, 5], [0, 0, -2.0, 5]]$
3) Forwards While Turning: $[[0.8, 0, 0.5, 8], [0.8, 0, 1.0, 5], [0.8, 0, -0.5, 5], [0.8, 0, -1.0, 5]]$
4) Lateral: $[[0, 0.4, 0, 8], [0, 0.8, 0, 5], [0, -0.4, 0, 5], [0, -0.8, 0, 5]]$

Note, the extra time allocated to the first command in each sequence is to account for the roughly two seconds of start up time between MuJoCo launching (the commands begin being sent to the robot immediately) and the robot being able to execute the commands. For evaluation we considered three "fixed" dynamic payloads, meaning the overall magnitude of the payload remained constant throughout deployment: 4.0 kg, 7.0 kg, and 10.0 kg. We also evaluated transporting a variable payload, which had the following schedule (notation - [payload mass (kg), duration (seconds)]): $[[6.5, 3.5], [7.5, 3.5], [8.5, 3.5], [3.0, 3.5], [10.0, 3.5], [4.5, 3.5]]$. The values of α, κ, λ_0, and k_0 (defined in the main text) for the BGF filter where set to 20.0, 1.2, 3.0, and 30.0 respectively in both simulation and physical robot experiments.

VI. Hardware Deployment

FLAP was deployed to robot hardware using a Nvidia Jetson Orin Developer Kit and a C++ based control stack. $\Phi(\cdot)$ and $\Psi(\cdot)$ are invoked sequentially in one thread operating at 0.5 kHz while the BGF adaptation law is run in a separate thread operating at 1 kHz. $\Psi(\cdot)$ is run on CPU.

References

[1] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[2] S. Yu, H. Hwang, T. M. Dang, J. Biswas, N. A. Giudice, S. I. Lee, and D. Kim, "Human-centered development of guide dog robots: Quiet and stable locomotion control," *arXiv preprint arXiv:2505.11808*, 2025.

[3] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and systems (RSS 2018)*, 2018.

[4] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *Advances in neural information processing systems*, vol. 33, pp. 5824–5836, 2020.

[5] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," *Advances in Neural Information Processing Systems*, 2017.

[6] G. Shi, W. Hönig, Y. Yue, and S.-J. Chung, "Neural-swarm: Decentralized close-proximity multirotor control using learned interactions," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[7] M. O'Connell, G. Shi, X. Shi, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural-fly enables rapid learning for agile flight in strong winds," *Science Robotics*, vol. 7, no. 66, 2022.