Solving practical multi-body dynamics problems using a single neural operator

Wenhao Ding\textsuperscript{a}, Qing He\textsuperscript{a,1}, Hanghang Tong\textsuperscript{a}, Qingjing Wang\textsuperscript{a}, and Ping Wang\textsuperscript{b}

\textsuperscript{a}MOE Key Laboratory of High-speed Railway Engineering, Southwest Jiaotong University, Chengdu, Sichuan, China; \textsuperscript{b}University of Illinois Urbana-Champaign, Champaign, USA

This manuscript was compiled on October 4, 2022

As a fundamental design tool in many engineering disciplines, multi-body dynamics (MBD) models a complex structure with a differential equation group containing multiple physical quantities. Engineers must constantly adjust structures at the design stage, which requires a highly efficient solver. The rise of deep learning technologies has offered new perspectives on MBD. Unfortunately, existing black-box models suffer from poor accuracy and robustness, while the advanced methodologies of single-output operator regression cannot deal with multiple quantities simultaneously. To address these challenges, we propose PINO-MBD, a deep learning framework for solving practical MBD problems based on the theory of physics-informed neural operator (PINO). PINO-MBD uses a single network for all quantities in a multi-body system, instead of training dozens, or even hundreds of networks as in the existing literature. We demonstrate the flexibility and feasibility of PINO-MBD for one toy example and two practical applications: vehicle-track coupled dynamics (VTCD) and reliability analysis of a four-storey building. The performance of VTCD indicates that our framework outperforms existing software and machine learning-based methods in terms of efficiency and precision, respectively. For the reliability analysis, PINO-MBD can provide higher-resolution results in less than a quarter of the time incurred when using the probability density evolution method (PDEM). This framework integrates mechanics and deep learning technologies and may reveal a new concept for MBD and probabilistic engineering.

Neural operator | Multi-body dynamics | Physics-informed deep learning | Probabilistic engineering

A branch of basic mechanics, multi-body dynamics (MBD) has evolved from its infancy in the 1970s into a fundamental design tool in many engineering areas (1). MBD transforms complex engineering structures, such as space vehicles, railway trains, and robotics into systems with multiple interconnected bodies, where the motion of each degree of freedom (DOF) is described using a partial differential equation (PDE) or ordinary differential equation (ODE). All the equations form a coupled differential equation group, the solution of which contains essential safety and performance information. With the mathematical tools obtained from the increasingly sophisticated theory of numerical integration (2), the current challenge is no longer merely solving the differential equation group, but rather the efficiency of this process. Computational efficiency is critical in many engineering problems using MBD. In a typical design project, engineers continuously adjust the structure and tune the structural parameters prior to manufacturing. Tens of thousands of differential equation groups must be solved until the performance of the design meets specific requirements (3). A higher computing power enables more design trials prior to the deadline, resulting in better product performance. Another example is the reliability assessment problems associated with buildings under seismic attack (4), bridges under traffic loads (5), and offshore structures under wave excitations (6, 7). In such cases, obtaining the damage probability field is the primary objective because serious casualties may occur when the structure fails. In the foreseeable future, it is unlikely that a general analytical method will be developed for large-scale and complex structures. Numerical and semi-analytical methods with statistical properties might remain mainstream, both of which require solving a large number of cases for sufficient probability resolution.

Unfortunately, mathematical approaches solve each equation group individually and therefore cannot learn from the solutions generated. The same applies to a physics-informed neural network (8) (PINN). Aiming to approximate the solution function of the governing PDE with a neural network, PINN along with its variants (9–11) are designed for a single PDE instead of parametric PDEs. Solving a new PDE with PINN requires retraining and often additional fine tuning (12), which substantially reduces its practicality in MBD problems. Therefore, to the best of our knowledge, all current machine-learning attempts to deal with MBD are based on classical deep neural networks (13, 14). Such black-box methods for MBD reportedly suffer from poor accuracy, interpretability, and weak generalization (13, 14) due to their lack of mechanics knowledge behind the data. Therefore, to fully benefit from the knowledge mechanics theory offers, we present PINO-MBD, a generalized physics-informed deep learning framework for solving MBD problems. The motivation for this approach stems from recent successes in deep learning for operator regression, such as ConvPDE-UQ (15), DeepONets (16), and neural operators (17). The concept of these approaches, and their physics-informed versions (18–20), is to learn the mapping between parameter and solution function spaces instead of approximating a single solution, such that a large number of predictions can be made with a simple forward pass once the network is trained. However, the characteristics of MBD pose additional technical challenges. The coupled differential equation group of a multi-body system contains multiple physical quantities, some of which are described with PDEs (flexible objects) and others (rigid objects) with ODEs. As our goal is to present a practical framework that competes with traditional numerical integration, all quantities are considered, forming a multi-output problem. Unfortunately, although some variants consider multiple inputs (21), most existing operator regression approaches are designed for a single physical quantity (single output). This is particularly true for DeepONet owing to its summation structure in the last layer (16). To account for the multivariate situation, multiple DeepONets must be pre-trained and then concatenated in a carefully designed system (22, 23).
The PINO-MBD addresses this challenge by taking light from both mechanics and deep learning techniques. Instead of training dozens or even hundreds of DeepONets to map all quantities (24), we only need one network by using the mode superposition method (25, 26) (MSUP) and a fully connected neural operator structure (27). An equation normalization (EN) technique is also proposed to deal with the imbalance of the MBD’s equation loss scales. This technique can improve the performance of derivatives in most cases without providing their ground truth.

Methodologically, our framework overcomes the limitations of both classic numerical integration and machine-learning approaches for MBD. Our approach, which can be applied to almost all MBD problems, offers an opportunity to increase the computing power by 3-4 orders of magnitude. In the following section, we first briefly introduce the workflow of PINO-MBD and elaborate on a few key techniques. Next, to demonstrate the flexibility of PINO-MBD and to validate its capability, we conducted experiments on one toy example and two practical applications: vehicle-track coupled dynamics (VTCD) and reliability analysis of a large-scale structure under seismic attack. In the following results, we show that this framework may be an important step forward in MBD and probabilistic engineering.

Overview of methods and workflow

In general, the PINO-MBD uses physical parameters and excitations as inputs to solve the dynamics of MBD systems. The overall workflow can be divided into two modules: a mechanical module and a deep learning module.

In the mechanical module (Fig. 1a), the PDE describing the motion of a flexible body is decomposed into multiple ODEs using the classic yet efficient MSUP (25, 26). Specifically, the solution field of the flexible body is approximated by the inner product of mode shape functions and mode amplitudes as follows:

\[ \{u^\ast\} = \sum_{i=1}^{n} \phi_i \ast \{u_i\}, \]  

where \( \{u^\ast\}, \phi, \{u\}, n, i \) represents the solution field, mode shape functions, mode amplitudes, number of modes used, and mode index, respectively. This operation allows us to process objects with complicated boundary geometries and many details because the governing 3D PDE of elasticity is transformed into multiple simplified 1D ODEs using prior physical knowledge (mode shape functions). These ODEs, along with those describing the motions of rigid bodies, form the coupled differential equation group of the system. We describe this differential equation group as follows:

\[ M(p, f, u, t) = 0, \quad \in D, \]  

where \( p \) and \( f \) denotes the physical parameter and system excitation with \( n_p \) and \( n_f \) elements respectively, \( u \) represents the solutions for all quantities in \( M \), and \( D \) is the bounded time domain.

With regard to the deep learning module, a neural network is trained to approximate the solution operator for Eq. (2). Let \( a = (p, f) \in A \subset \mathbb{R}^{n_p+n_f} \) be the variable parameter set for the MBD system, and \( u \in U \) be the solution set with \( n_{do} \) elements. The deep learning module then aims to build an approximation \( G_{\theta} \) of the solution operator \( G \) for the coupled differential equation group \( M \), which is expressed as follows:

\[ G_{\theta} : A \rightarrow U, \quad \theta \in \mathbb{R}^{n_p+n_f}, \]  

with parameter configurations from finite-dimensional space \( A \subset \mathbb{R}^{n_p+n_f} \). We used a Fourier neural operator (27) based on graph kernel network theory (28) as the backbone for this module. A fully connected network is connected subsequently, allowing the module to deal with multiple equations in Eq. (2) simultaneously, in which only the solutions are output, and their derivatives are computed through an extra numerical differential operation. The complete solution and derivative fields can then be easily recovered using Eq. (1). The loss function is constructed with the data loss \( L_{data} \), equation loss \( L_{eq} \), direct derivative loss \( L_{ddea} \) and virtual equation loss \( L_{veq} \). The four components can be selected according to the application scenario, and their detailed definitions are listed in Subsection PINO. The application of \( L_{ddea} \) is driven by the fact that the engineering community is generally more concerned about derivatives than solutions. For example, in nature sound is generally related to the vibration speed (1st derivative), while human comfort is related to acceleration (2nd derivative). Furthermore, we also propose an EN technique to enhance \( L_{eq} \) as an alternative because using \( L_{ddea} \) leads to more GPU occupation.

Results

Application demonstration on a toy example. To demonstrate the flexibility of PINO-MBD, and to validate its capability to guarantee the accuracy for both solutions and derivatives, a typical multi-body structure is used as a toy example. This example consists of two flexible rubber components with different elastic modules, and six rigid steel marbles, all of which are connected to each other via a spring-dashpot mechanism. In the mechanical module, 15 and 10 modes were considered for the upper and lower rubber components respectively. We trained the PINO-MBD to output solutions for all 43 DOFs, considering variable marble mass, spring-dashpot properties,
Fig. 1. Overall workflow of the PINO-MBD. a, Mechanical module. The governing PDE for an elastic flexible component is decomposed into multiple ODEs using the mode shape functions. Multiple components are represented by two stacked ones in the figure. All ODEs together form a coupled differential equation group for the entire multi-body system. b, EN technique. This technique simulates the learning process by adding perturbations into different equations, and then the weight for each equation in each data pair can be calculated accordingly. c, Deep learning module. The output solutions and their derivatives are fed into the physical equations with weights configured through EN to compute the equation losses, making the network physics-informed. Finally, the solution and derivative fields can be recovered with Eq. (1).

Fig. 2. PINO-MBD performance for the toy example. a, Recovered 3D solution and derivative fields for the flexible components. The results for displacement fields are visualized with a scale factor of 50. b, Statistical results of equation losses for different ODEs with a 2% perturbation error added to their ground truth. c, Mesh-independence verification of PINO-MBD. Recovered solution and derivative fields from two different mesh grids are compared. The annular areas in both grids are meshed with a finer size.
and stochastic excitations. All data (solutions and derivatives) were first normalized to the dataset (with standardization) before their relative L2 losses ($rLSE$) were calculated to evaluate the performance of the PINO-MBD.

After 300 epochs, the 1D responses of mode amplitudes for which we intend to compete using the PINO-MBD. These practical demands have facilitated the development of the vehicle-track system (10 DOFs for the vehicle system and 4 additional training, which differentiates PINO-MBD from the mode shape vectors to recover the responses, requiring no equation losses indicates that the gradients PINO-MBD harvested from different equations and data pairs were uneven during training. We discuss the effect of this gradient imbalance phenomenon on the training quality in Subsection Analysis on training quality.

Finally, the mesh-independence of PINO-MBD is verified in Fig. 2c. The annual areas in both grids were meshed to a finer size. The trained PINO-MBD can directly use new mode shape vectors to recover the responses, requiring no additional training, which differentiates PINO-MBD from other mesh-dependent deep learning methods (29–31). In practice, one can simply use element shape functions to obtain responses at desired positions, just like the finite element method (32).

Practical application of VTCD. As major transportation arteries in many countries, railways play an important role in social and economic development, which has stimulated the study of VTCD (33, 34). In the past 20 years, the theory of VTCD has been verified through numerous site experiments (35). Currently, VTCD is widely used in many aspects of railway engineering, including integrated designs of modern rolling stocks and track structures, safety evaluation of new lines before operation, and engineering education. These practical demands have facilitated the development of corresponding software modules, such as Simpack® and UM®, with which we intend to compete using the PINO-MBD. In this example, all properties of the running vehicle were considered to be variable, including suspension parameters, the distance between bogies, the distance between wheelsets, and running speed. For the track structure, the stiffness and damping of the rail pads, along with stochastic rail irregularities on the surface, were also variable. Once trained, PINO-MBD can instantly output dynamic responses for the vehicle-track system (10 DOFs for the vehicle system and 4 DOFs for rail deformations under wheelsets) considering arbitrary excitations and parameter configurations. As shown in Fig. 3a, the output results (V2 in Table 1) achieved good accuracy. Specifically, system displacement, velocity, and acceleration achieved 4.80%, 4.23%, and 4.30% $rLSE$ after 300 epochs.

Compared with numerical integration, the main advantage of PINO-MBD lies in its efficiency. For a 5 s dynamic response of a train running at 350 km/h, UM® and Simpack® take 18 and 23 s, respectively. In contrast, performing a forward prediction of 10,000 cases with PINO-MBD takes less than 0.02 s, indicating a magnitude improvement of three orders in computational efficiency. Furthermore, solving the VTCD with PINO-MBD only requires sufficient memory on the computer, whereas numerical integration methods are notorious for requiring a considerable amount of CPU processing power.

On the other hand, PINO-MBD’s performance is more elegant in terms of both functionality and precision than the existing deep-learning based approach MBSNet (14). As shown in Fig. 3d, PINO-MBD outperformed MBSNet by at least one order of magnitude in accuracy for most DOFs. In addition, PINO-MBD not only provides solutions but also guarantees the accuracy of their derivatives, which is much more challenging than just providing all results, as is done by MBSNet.

Practical application of reliability analysis to a large-scale structure under seismic attack. To further demonstrate the potential of the PINO-MBD for probabilistic engineering, we performed a reliability assessment for a four-storey building modeled with 67881 solid elements (135426 DOFs). The task of a probabilistic engineer is to evaluate the aseismic reliability by solving the probability characteristics of its dynamic response under a 3D stochastic seismic attack. This can be achieved using the Monte Carlo method (MCM), that is, solving a large number of cases considering the stochastic nature of earthquakes, and then statistically obtaining the damage probability. However, this approach is practically infeasible owing to the power-hungry nature of MCM. For this particular building, it took approximately 1 h to solve a 5 s seismic response on an Intel Core i9-11900K CPU, implying that it would take more than a year to solve 50,000 cases with five computers.

Bypassing the computational consumption of the MCM is one of the main research goals in the field of probabilistic engineering. Since the study of Brownian motion by Einstein (1905), many approaches to this problem have been proposed, such as the Liouville equation, FPK equation, and Dostupov-Pugachev equation (37). We used the probability density evolution method (PDEM) (36–38) (workflow listed in subsection PDEM) as the baseline for the PINO-MBD to compete with, considering its popularity and successful applications on many engineering structures, such as buildings (39), slopes (40), tie-back walls (41), and bridges (42, 43). Fundamentally, the PDEM produces the probability density function (PDF) results of the desired response over time by superimposing the evolution results of a smaller number of cases (compared to MCM). In this case, we used 499 cases generated with uniform design (47) for the PDEM, which is sufficient compared with similar research. The evolution result of each case is obtained by solving a 1D convection equation that describes the flow of the deterministic responses
Fig. 3. | Application demonstration of VTCD. a, Schematic for the train-track coupled system. b, Performance of PINO-MBD on the dynamic responses of the vehicle-track coupled system. The solutions and their derivatives are visualized for 4 out of 14 DOFs. Definitions for the three cases (V2, V3, and V4) demonstrated here are listed in Table 1. c, Statistical results of equation losses for different ODEs with a 2% perturbation error added to their ground truth. d, Performance comparison among PINO-MBD trained with EN, PINO-MBD trained without EN, and MBSNet. The performance data of MBSNet was extracted from (14).
Fig. 4. Application demonstration of reliability analysis on a large-scale structure. 

a, Recovered displacement field of a four-storey building under seismic attack. 

b, The mean field of maximum main stress is shown on the left and the damage probability field is on the right. 

We refer to the right edge area with higher main stress as the belly area. 

d, Sliced observations of probability density evolution surfaces at different times. The observation positions on the time axis are visualized as yellow dash lines in sub-figure Fig. 4c. $dP^*$ denotes the damage probability at the specific time.
in the probability space.

We trained the PINO-MBD to learn the mapping between seismic excitations and the structural response on a small dataset with 150 data pairs. Subsequently, we test the PINO-MBD perform fast forward prediction under 49,999 stochastic 3D seismic waves, and subsequently recovered the stress field to obtain the damage probability statistically. This process shares the same philosophy as MCM, except that the computational cost is significantly reduced with the help of the PINO-MBD. Fig. 4a visualizes the recovered displacement fields generated with the output mode amplitude responses, in which 200 modes were considered. The PINO-MBD prediction achieved good accuracy compared with the ground truth, with a $r_{LSE}$ of 4.81% after 300 epochs.

The probability results statistically obtained from 49,999 predictions using the trained PINO-MBD are shown in Fig. 4b-d. Through these results, we observe that PINO-MBD outperforms PDEM in the following aspects.

1). PINO-MBD allows engineers to observe the entire probability field of the structure.

Looking at Fig. 4b, the hotter areas on the left have larger mean values of tensile stress over a total of 49,999 data pairs. On the right, $dp^*$ denotes the probability of concrete cracking and fracturing, that is, the probability that the main stress exceeds the tensile strength (1.71 MPa) during an earthquake. Using this figure, engineers can easily determine the vulnerable areas of the structure and the damage probability at different locations, providing decision-makers with a basis for evaluating and optimizing designs. In contrast, it is impossible to solve the probability evolution results for all DOFs using the PDEM because solving the 1D convection equation is time-consuming. Therefore, PDEM can only provide results for a limited number of observation positions; that is, Fig. 4b cannot be generated with the PDEM.

2). PINO-MBD provides probability evolution results with a higher resolution.

As shown in Fig. 4c, the probability evolution results produced by the PDEM are rough and uneven compared with those of PINO-MBD. A closer examination of Fig. 4d reveals many isolated small probability hills in the belly area from the results generated with the PDEM. This phenomenon has been reported in many papers (40, 44, 45), but to the best of our knowledge, it has not been discussed. The high-resolution results of PINO-MBD indicate that the roughness of the evolution results and the isolated hills in the belly area are not inherent properties of the probabilistic results per se, but rather a manifestation of the PDEM’s insufficient resolution. This advantage of PINO-MBD has practical implications for reliability assessment. Considering Fig. 4d as an example, at $T=4.0$ s, $dp^*$ calculated by the PINO-MBD is 0.03% whereas that calculated by the PDEM is less than 0.01%, suggesting that the PDEM can be unconservative.

3). The PINO-MBD was three times more efficient.

PINO-MBD took only 37 h on five threads, including generating training data (30 h), training (3 h), and post-processing (4 h). Contrastingly, PDEM took 149 h, including computing 499 cases (100 h), evolving the results (48 h), and post-processing (1 h). PINO-MBD demonstrated a 302.7% increase in computational efficiency.

Analysis on training quality. To further understand the key elements affecting training quality, we conducted a series of experiments, considering different network properties and loss function compositions, as shown in Table 1 and Fig. 5. First, we note the importance of embedding physical knowledge for MBD. A bare neural operator as the backbone generally demonstrates slightly better performance on solutions but deteriorates dramatically on derivatives (T1 vs T6, V2 vs V4, R2 vs R4, Fig. 3a). These results echo the finding in (18), emphasizing the necessity of embedding physics because the engineering community often values derivatives more than solutions.

Next, the benefits of using EN were noted by comparing T1 with T5, V1 with V3, and R2 with R3. EN enhances the performance of the PINO-MBD on derivatives at the cost of a slight deterioration of solutions. For VTCD, EN offered 1.21% and 9.89% improvement on 1st and 2nd derivatives, but only a 3.1% improvement on 2nd derivatives was observed for the toy example. We reason that this is because the imbalance situation for VTCD is much worse, as observed from Fig. 2c and Fig. 3c. Without EN, the gradients harvested from some ODEs (those with larger equation losses when the perturbation is added) are much larger than others, prohibiting PINO-MBD from learning mappings for all DOFs uniformly. For applications with higher precision requirements, direct $L_{dd}$ is still necessary (which is generally wasted in existing studies) at the cost of more GPU occupation and generating ground truth for derivatives.

Finally, the advantages of using virtual loss can be quantified by comparing T1 to T3, and T2 to T4. The benefits of using virtual loss $L_{veq}$ are limited, possibly because their training datasets are already large enough. This raises the question of whether PINO-MBD can be trained without providing any ground truth data (only with $L_{veq}$). We tested this idea on a simple multi-body system, as shown in Fig. 6. Stochastic vibration loads were applied to three steel marbles, and fixed constraints were imposed on the perimeter of the steel plate at the bottom. Currently, the results obtained from the framework trained with only $L_{veq}$ are unsatisfactory. However, $r_{LSE}$ for the solutions can be reduced to less than 6% when boundary constraints $L_{dd(+)}$ are provided, as shown in Fig. 7.

We envision that general non-data training strategies can be realized in future work by taking light from the existing research on PINN.

Discussion

This study provides a proof of concept that physics-informed deep learning can successfully and efficiently regress operators for MBD. Owing to the ever-increasing computing power in the last decade, engineering corporations and design institutes have accumulated huge simulation data but have conducted limited mining. By taking light from both mechanics and deep learning technologies, PINO-MBD allows engineers to model complex, practical systems with complicated boundary conditions and numerous details. More importantly, this can be achieved with a single network, instead of dozens or even hundreds as in the existing literature. This study, in turn, leads to a new concept that design institutes can train this framework for their specialized engineering objects to handle projects with different requirements, rather than modeling
Fig. 5. Relative $L^2$-E performance for different loss function combinations. $rLSE$ for MBD solutions and their derivatives are visualized at 25%, 50%, and 100% of the training process. The first to third rows correspond to the toy example, VTCD, and the reliability assessment on the four-storey building. Each color for the boxes represents an individual case, corresponding to Table 1.

Fig. 6. Schematic of the simple mechanical structure.

each project individually. In the future, we anticipate the development of software with GUI interfaces that will compete with the existing MBD software. To demonstrate its flexibility and feasibility, we provided a general toy example and two practical applications. For VTCD, PINO-MBD outperformed the commercial software and existing deep learning approaches in terms of efficiency and accuracy, respectively. We believe the better performance of the PINO-MBD compared to time series prediction methods such as CNN-LSTM (used in MBSNet) should be attributed to the stability of mechanical system responses in the frequency domain. In fact, finding the mappings between input and output functions in the frequency domain coincides with the idea of transfer function approaches (2). However, it is notoriously difficult to obtain transfer functions for complex structures using pure mechanical derivations, and our research has proven that deep learning can do better in this regard. Moreover, although the main target of our study is MBD, this framework can also be applied to other operator regression problems dealing with coupled differential equation groups. Different from existing approaches such as DeepMcCNet (22, 23), we do not train multiple neural networks in advance. Instead, a single neural operator is used to output responses for all coupled physical quantities simultaneously. We believe this idea may provide a more concise approach to other disciplines such as multiphysics, which we will investigate in the future.

Furthermore, we demonstrated the potential of this framework for reliability assessments. We used this framework as a magnifying glass to observe the probabilistic characteristics of the dynamic structural responses. Thanks to the higher resolution brought by PINO-MBD, we found that the roughness and isolated probability hills are, in fact, caused by the insufficient resolution of the results solved from the probabilistic equation (PDEM), which has been observed in many studies. This may provide a new perspective for research in other disciplines since probabilistic and stochastic equations are widely used in physics (49, 50), biology (51), and finance (52). In conclusion, this study integrates MBD theory and deep learning-based operator regression technologies and enables engineers to solve complex systems efficiently and accurately. Although only a few examples are demonstrated here, we expect PINO-MBD to be broadly ap-
Materials and Methods

**MSUP.** The MSUP is a classic yet powerful technique for dynamics, in which the dynamic response of a structure is approximated by the superposition of a small number of its eigenmodes, as shown in Eq. (4).

\[
\{u^*_i\} \sum_{i=1}^{n} \phi_i \ast \{u_i\},
\]

\[
\{u^*_i\} = \text{solution field (displacement)}
\]

\[
\{\phi\} = \text{mode shape function}
\]

\[
\{u\} = \text{mode amplitude}
\]

\[
\{n\} = \text{number of modes used}
\]

\[
\{i\} = \text{mode index}
\]

Here, the mode shape function \(\phi_i\) is an inherent property of the flexible object itself, which is mesh-independent and only related to the material and geometry. The analytical \(\phi\) can be obtained through an analytical derivation for simple mechanical structures. For example, the rail structure in the VTCD was modeled with an Euler beam, whose analytical mode shape functions are described as follows:

\[
Z_k(x) = \sqrt{\frac{2}{m_r l^2}} \sin \frac{k \pi x}{l},
\]

where \(m_r\) and \(l\) denotes the mass per meter and total length, respectively. For more complicated objects, we used finite element method (FEM) to spatially approximate the elastic PDE and compute the numerical \(\phi\). Assume that the equations for an object discretized with FEM are written in matrix form, as follows:

\[
M \ddot{u} + C \dot{u} + Ku = f(t),
\]

where \(M\) denotes the mass matrix, \(C\) denotes the damping matrix, and \(K\) denotes the stiffness matrix. The DOFs are placed in the column vector \(u\) and the forces are in \(f(t)\). The eigenfrequencies and corresponding mode shapes can be solved using the eigenvalue equation in Eq. (7).

\[
(-\omega^2M + K)u = 0, \quad u \neq 0
\]

\[
U^T M U \ddot{q} + U^T C U \dot{q} + U^T K U q = U^T f(t)
\]

\[
C = \alpha M + \beta K
\]

\[
\mu \ddot{q} + \Omega q = r(t)
\]

\[
\gamma_i = U(:,i)^T M D U, \quad M_{eff} = \sum_{i=1}^{n} \gamma_i^2, \quad M_{eff} \geq 95%
\]

The mode shapes (eigenmodes) are the spatial discretization of the inherent \(\phi\), and only a small number \(n\) of eigenfrequencies need to be computed. It is convenient to place the eigenmodes in a rectangular matrix \(U\), where each column contains an eigenmode. Eq. (6) can be transformed into Eq. (8) after a left multiplication by \(U^T\), and finally into Eq. (10), using

### Table 1. Performance analysis under different loss function combinations. (average \(rLSE\) over multiple training).

| Index | Network | Loss function | \(rLSE (\%)\) | Solutions | \(1^{st} Des\) | \(2^{nd} Des\) | Average |
|-------|---------|---------------|---------------|-----------|--------------|--------------|---------|
|       | PI EN   | \(L_{eq}\)    | \(L_{dde}\)  | \(L_{veq}\) |             |              |         |
| Toy example |       |               |               |           |             |              |         |
| T1    | ✓ ✓     | ✓             | *             |           | 3.85        | 4.28         | 6.14    | 4.76 |
| T2    | ✓ ✓     | ✓             | ✓             |           | 3.54        | 3.73         | 4.47    | 3.91 |
| T3    | ✓ ✓     | ✓             | *             |           | 3.96        | 4.39         | 6.03    | 4.79 |
| T4    | ✓ ✓     | ✓             | *             |           | 3.60        | 3.80         | 4.46    | 3.95 |
| T5    | ✓ ✓     | ✓             | *             |           | 3.34        | 3.86         | 9.24    | 5.49 |
| T6    | ✓ ✓     | ✓             | *             |           | 3.37        | 3.79         | 7.81    | 4.99 |
| T7    | ✓ ✓     | ✓             | *             |           | 3.37        | 4.28         | 39.51   | 15.72 |
| Vehicle-track coupled dynamics |       |               |               |           |             |              |         |
| V1    | ✓ ✓     | ✓             | *             |           | 4.22        | 2.42         | 3.42    | 3.35 |
| V2    | ✓ ✓     | ✓             | *             |           | 4.80        | 3.56         | 5.25    | 4.54 |
| V3    | ✓ ✓     | ✓             | *             |           | 4.12        | 4.77         | 15.14   | 8.00 |
| V4    | ✓ ✓     | ✓             | *             |           | 3.14        | 12.32        | 2584.13 | 866.53 |
| Reliability assessment for a 4-storey building |       |               |               |           |             |              |         |
| R1    | ✓ ✓     | ✓             | *             |           | 4.81        | 5.13         | 6.37    | 5.44 |
| R2    | ✓ ✓     | ✓             | *             |           | 4.47        | 10.39        | 25.12   | 13.33 |
| R3    | ✓ ✓     | ✓             | *             |           | 12.75       | 26.03        | 40.97   | 26.59 |
| R4    | ✓ ✓     | ✓             | *             |           | 2.62        | 23.51        | 2116.41 | 714.18 |

In this table, PI marked with ✓ indicates the trained model is physics-informed, while EN marked with ✓ means equation normalization technique was used. For each experiment, loss components marked with * were used during training. For \(L_{dde}\) marked with ✓, we only provided the ground truth of derivatives for a small period of time (0.025s) on the edges of the time domain as a boundary constraint. \(Des\) stands for derivatives, and experiments with colored backgrounds are visualized in Fig. 5. All data were normalized on corresponding datasets before their \(rLSE\) were computed.
the Rayleigh damping model (where \(\alpha\) and \(\beta\) are material-dependent constants) in Eq. (9). At this stage, the number of governing equations is reduced from the system DOF number in Eq. (6) into \(n\) in Eq. (10). To retain accuracy, a sufficient \(n\) must be considered to ensure a sufficient effective mass \(M_{eff}\) in Eq. (11), where \(D\) is an assumed unit-displacement vector. In this study, we primarily used ABAQUS to mesh flexible objects and ANSYS to compute the mode shapes. Specifically, we used 15 and 10 modes for the upper and lower flexible bodies in the toy example, respectively, and 200 modes for the four-storey building in the reliability assessment, all satisfying Eq. (11).

Physics-informed neural operator (PINO). In the deep learning module, we used PINO (20) with a Fourier neural operator (FNO) core as the backbone of the framework. The data features of the system excitations in the frequency domain were extracted through convolution operations in Fourier convolution layers considering different system parameter configurations. These features were then fed into the fully connected layers to be combined to output the dynamic responses for different DOFs in the MBD system. Fundamentally, the FNO aims to determine the mappings between the input and output data in the frequency domain. The convolution operations were performed with a fast Fourier transformation (FFT). The fully connected layers connected subsequently allow the deep learning module to form different feature combinations for multiple outputs. This workflow allows PINO-MBD to use a single network for all DOFs in the MBD system, thus avoiding training multiple networks as in (22, 23) (which is not practical for MBD with hundreds of DOFs). We let PINO-MBD output solutions for 43 DOFs in the toy example (15 and 10 mode amplitudes for the upper and lower flexible objects, three rigid amplitudes in the reliability assessment application. Regarding the dataset size, the train and test set sizes are 1,200 and 1,000 for the toy example, 10,000 and 2,000 for VTCD, and 150 and 25 for the reliability analysis.

During training, the PINO-MBD harvests gradients from four types of loss functions: data loss \(L_{data}\), equation loss \(L_{eq}\), direct derivative loss \(L_{dde}\), and virtual equation loss \(L_{veq}\).

\[
\begin{align*}
L_{data}(a, \mathcal{G}(a)) &= \mathbb{E}_{a \sim u} \left[ \| u - \mathcal{G}(a) \|_2^2 \right] \\
L_{eq} &= \mathbb{E}_{a \sim u} \left[ \| M(a, \mathcal{G}(a)) \|_2^2 \right] \\
L_{dde} &= \mathbb{E}_{a \sim u} \left[ \frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} \right] - \left( \frac{\partial \mathcal{G}(a)}{\partial t} + \frac{\partial^2 \mathcal{G}(a)}{\partial x^2} \right) \right\|^2_{L_2} \\
L_{veq} &= \mathbb{E}_{a \sim u} \left[ \left( \frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} \right) - \left( \frac{\partial \mathcal{G}(a)}{\partial t} + \frac{\partial^2 \mathcal{G}(a)}{\partial x^2} \right) \right]^2_{L_2}
\end{align*}
\]

We assume a dataset \(\{a_j, u_j\}_{j=1}^N\) is available, where \(a_j \sim \mu\) are parameter configurations sampled from some multi-dimensional distribution \(\mu\) supported on \(A\). Before training, data normalization is performed for \(a_j\) and \(u_j\), ensuring that data losses \(L_{data}\) obtained from different DOFs are comparable. Equation loss \(L_{eq}\) in Eq. (12) is defined as the average squared norm value over the time
domain in $A$. During the training process, PINO-MBD outputs only the solutions for the MBD system, and their derivatives are computed through a differential operation after being renormalized. We used a simple numerical difference operation in the time domain to achieve this, considering its comprehensive advantages of efficiency, low GPU occupancy and stability. These data were then fed into the differential equation group $M$ to generate the solution loss $L_{eq}$. $L_{eq}$ shares the same expression as $L_{ode}$, except that it is sampled from an additional dataset $\{a_i\}_{k=1}^M$, where the parameter configurations are provided without the corresponding ground truth for the solutions. In the case of small datasets, $L_{eq}$ enables PINO-MBD to obtain equation losses on a larger dataset. This hardly increases the training cost since no corresponding ground truth needs to be generated. Finally, $L_{ode}$ was computed by directly providing PINO-MBD with the ground truth of the $1^{st}$ and $2^{nd}$ derivatives. Undoubtedly, using $L_{ode}$ positively affects the accuracy of derivatives by taking advantage of the ground truth data for derivatives, which is generally wasted in existing research. This is especially true for derivative-insensitive equations, where the accuracy of derivatives has little effect on their equation losses. However, although generating the ground truth data for derivatives does not increase the cost significantly (most numerical integration algorithms can provide both their solutions and their derivatives at the same time), using $L_{ode}$ requires more GPU occupation, thus slowing down the training process. Furthermore, for some cases (denoted by + in Table 1), we only computed $L_{ode}$ for a small period of time on the edges of the time domain. This will help PINO-MBD impose sufficient boundary constraints on the time domain to improve the performance without significantly increasing the training cost. The four loss functions were balanced with weights $\omega_i$, $(i = 1 \sim 4)$ controlled by the GradNorm technique (46), and different combinations can be selected for different purposes. For example, users looking for precision may tend to use $L_{ode}$, whereas those seeking to reduce training costs may tend to use $L_{eq}$.

The EN technique. For common operator regression problems with a single PDE, applying a manually determined weight for each type of loss function is sufficient. However, this does not apply to MBD systems because the differential equation group $M$ contains $n_{def}$ ODEs, each describing a unique physical motion. During the training process, the scales of the equation losses for different ODEs in $M$ can vary significantly when their data losses are similar. Because of this MBD-unique imbalance phenomenon, most equation losses harvested by PINO-MBD are only from sensitive ODEs, that is, ODEs in which the accuracy of the solutions has a significant influence on their equation losses. However, the sensitivity of an equation is merely a mathematical feature and cannot represent the importance of the physical quantity it describes. To address this, we propose an EN technique that fully utilizes prior physics knowledge. Specifically, EN generates a weight $\lambda$ for each ODE in each data pair, which can be used to normalize their equation losses during training. These weights allow the PINO-MBD to approximate different ODEs uniformly during training, thus improving the overall performance. The full EN algorithm is summarized in Algorithm 1. Here, $\sigma(x)$ represents the standard deviation of $x$, and $\varepsilon(a,b)$ generates a random sequence following a uniform distribution on $(-a,a)$, with the same dimension as $b$, $\max^T |M| \lambda$ computes the extreme value in the time domain for each ODE in $M$. $r$ is a predefined constant representing the acceptable error level, which is set as 2% in this study. During training, $\lambda_{ij}$ is allocated to each ODE in each data pair, which enables their equation losses to be normalized to approximate $r$ when their data losses are close to $r$. From another perspective, adding perturbation to the ground truth and computing the corresponding equation losses is, in fact, a simulation of the learning process. The degree of difference in the equation losses for different ODEs indicates the severity of the imbalance (Fig. 2c, Fig. 3c).

| Algorithm 1 | Compute weight $\lambda_{ij}$ for the $j^{th}$ ODE in the $i^{th}$ data pair |
|--------------|---------------------------------------------------------------|
| 1: $i = 1 \rightarrow N$ do | |
| 2: Extract data pair $\{a_i, u_i\}$ | |
| 3: $j = 1 \rightarrow n_{def}$ do | |
| 4: $u = u_i(\cdot, j)$ | |
| 5: $p_0 \leftarrow \varepsilon(r \ast \sigma(u_i), u_i), p_0 \leftarrow \varepsilon(r \ast \sigma(u_i), \dot{u}_i), p_0 \leftarrow \varepsilon(r \ast \sigma(u_i), \ddot{u}_i)$ | |
| 6: $s_0(\cdot, j) \leftarrow u + p_0, s_1(\cdot, j) \leftarrow \frac{\partial u}{\partial x} + p_1, s_2(\cdot, j) \leftarrow \frac{\partial^2 u}{\partial x^2} + p_2,$ | |
| 7: $L = \max^T |M(a_s, s_0, s_1, s_2)|$ | |
| 8: $j = 1 \rightarrow n_{def}$ do | |
| 9: $\lambda_{ij} = \frac{r}{L_{ij}}$ | |

PDEM. The PDEM was developed by Li and Chen at the beginning of this century based on the principle of probability preservation (36–38). The past two decades have witnessed its successful applications on many stochastic engineering problems, particularly seismic analyses. Therefore, we used the PDEM in this work to perform a reliability analysis for a four-storey building as a baseline for PINO-MBD to compete with. The governing equation of the PDEM is written as:

$$\frac{\partial p_X}{\partial t}(x,a,t) + X(a,t) \frac{\partial p_X}{\partial x}(x,a,t) = 0$$

and the probability density function of $X(t)$ reads

$$p_X(x,t) = \int_A p_X(x,a,t) \, da$$

where $X$ denotes the target physical quantity; $a$ is the random parameters involved; $p_X(x,a,t)$ is the joint probability density function of $(X,a)$; $A$ is the random parameters space; $p_X(x,t)$ represents the PDF of $X(t)$.

Eq. (13) is a 1D convection PDE bridging the joint PDF of $(X,a)$ and the corresponding velocity $X(a,t)$. Furthermore, the PDF of a certified response is completely determined by its own velocity, regardless of other response quantities. Fundamentally, the PDEM generates a number of 2D $p_X(x,t)$ results from their deterministic 1D system responses $X(a,t)$ using Eq. (13). These results are superimposed to approximate the desired $p_X(x,t)$ as a discrete realization of Eq. (14). The full PDEM algorithm is summarized in Algorithm 2.

In the reliability assessment problem, the 3D seismic excitations applied to the foundation of the building were
Algorithm 2 Computing $p_X(x, t)$ with PDEM
1: Pick out representative points $a_q (q = 1, 2, \ldots, N_{sel})$ in $A$.
2: for $q = 1 \to N_{sel}$ do
3: Carry out deterministic time integration on the 4-storey building, yielding the velocity $\dot{X}(a_q, t)$.
4: Introducing the evaluated $\dot{X}(a_q)$, solve Eq. (13) with the finite difference method, yielding $p_{X_{a_q}}(x, t)$.
5: $p_X(x, t) = \frac{1}{N_{sel}} \sum_{q=1}^{N_{sel}} p_{X_{a_q}}(x, t)$

generated using the Kanai-Tajimi spectrum. The earthquake acceleration in each direction was generated using a spectrum method with one random phase (48), forming a 3D random parameter space $A$ for this example. We used a uniform design (47) based on number theory to extract 499 representative points in $A$ uniformly, which we believe is sufficient compared with similar research items.

$X$ in Eq. (13) can be any desired physical quantity from displacement, velocity, and acceleration to the main stress. However, it should be emphasized that solving the 1D convection PDE in Eq. (13) is time-consuming, although multiple quantities can be solved in parallel. Therefore, the PDEM can only provide the evolution results of PDF ($p(x, t)$) for a limited number of $X$. In this study, we focused on the main stress of concrete for a few locations on the first floor, as shown in Fig. 4b.

ACKNOWLEDGMENTS. XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX XXX X