PHYSICS-BASED INTERPOLATION FOR ANIMATION USING PHYSICS-INFORMED NEURAL NETWORKS

An Undergraduate Research Scholars Thesis

by

CARLOS ALVAREZ DEL CASTILLO SALEH

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: Dr. Shinjiro Sueda

May 2023

Major: Computer Science

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

I, Carlos Alvarez del Castillo Saleh, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with my Faculty Research Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

ABSTRACT

Physics-Based Interpolation for Animation Using Physics-Informed Neural Networks

Carlos Alvarez del Castillo Saleh
Department of Computer Science and Engineering
Texas A&M University

Faculty Research Advisor: Dr. Shinjiro Sueda
Department of Computer Science and Engineering
Texas A&M University

Keyframe interpolation is a fundamental technique used in computer animation. In this technique, an animator specifies keyframes and has a computer program generate the in-betweens by interpolating between them. However, most common forms of interpolation use a non-physics-based path between keyframes to decide where the in-betweens should be drawn. When animating physical behavior, interpolating along a physics-based path of motion could create more realistic animations while reducing the number of keyframes that must be specified by the animator. Physically Informed Neural Networks (PINNs) are neural networks used to solve problems involving partial differential equations by directly incorporating information about those equations into the network. Because prior knowledge of the functions they are trying to model is incorporated into the network, one advantage of PINNs is they require less data to train than neural networks trained using only data points. This research trains PINNs to interpolate the path of an animated object, specifically a mass attached to a two-dimensional spring, in a physics-based manner while closely meeting the constraints imposed on the object's position and/or velocity at keyframes. In addition, this project explores how modifying the

hyperparameters and boundary conditions used to train the PINN affects the interpolation and the efficiency of the training process.

ACKNOWLEDGMENTS

Contributors

I would like to thank my faculty advisor, Dr. Shinjiro Sueda, for his guidance and support throughout the course of this research.

I would also like to thank Vinesh Ravuri, Bethany Witemeyer, and Ryan Zesch for their help and support.

The PINNs used in this paper were created and trained using the PyTorch library. The graphs in this paper were created using the Matplotlib library and the colorline interface from this Jupyter notebook: https://nbviewer.org/github/dpsanders/matplotlib-examples/blob/master/colorline.ipynb.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

This project received no funding.

1. INTRODUCTION

1.1 Keyframe Animation

Animations are made up of frames, a series of still images that give the illusion of motion when played in sequence. To create animations with the assistance of a computer, a technique called keyframe animation is often used. In this technique, animators specify keyframes, a series of poses that represent significant moments in the animation. A computer program then interpolates the keyframes to automatically generate in-between frames [1]. For example, when animating a falling ball, an animator may specify a keyframe depicting the ball at its highest point, and another keyframe depicting the ball at its lowest point. The in-between frames of the ball falling from its highest to its lowest point would then be automatically generated by the computer program. This is demonstrated in Figure 1.

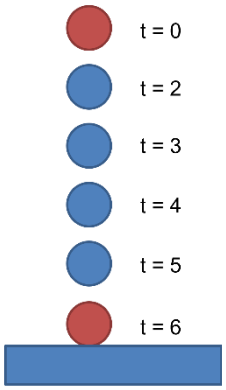

Figure 1: A conceptual example of the keyframing process for an animation of a falling ball. The keyframes are colored in red and would be specified by an animator. The in-between frames are colored blue and would be automatically generated by a computer program.

The main advantage of keyframe animation is that it allows animators to maintain a high degree of control over their animations while being simple to use. That said, most common forms of keyframe interpolation use non-physics-based paths to determine the placement of in-

between frames. As a result, the resulting animations may appear unnatural or lack realism. To address these issues while using this method, it is possible to add more keyframes or adjust the timing of the animation, but this can become very time-consuming as the complexity of the animation increases.

1.2 Physics-Based Animation

Physics-based animation encompasses a large variety of animation techniques. These techniques use physical laws and constraints to create motion, often by simulating the objects being animated [2]. For example, when animating a ball falling, an animator may specify the initial position and properties of the ball on the first frame of the animation. The computer will then run a simulation which will calculate the effects of gravity on the ball and determine where to draw it in the subsequent frames. One advantage of physics-based methods is that the animation generated by these methods inherently follows physical laws and can thus look very realistic. Moreover, unlike keyframe animation, physics-based animation doesn't need keyframes. As a result, this can save significant amounts of time when creating complex animations. However, the animator often no longer has the same level of control over the animation that keyframing provides even though there may be ways to interact with the simulation to create the desired animation. In cases where the animator is more concerned with achieving an overall effect rather than controlling every aspect of the animation, this is acceptable. For example, when animating a splash, it is unlikely the animator cares exactly where every drop of water is at every point in time if the splash looks good. For other situations, the animator may want to animate an object with a physics-based trajectory while ensuring the object passes through specific points at specific times. In this case, it may be complicated for the

animator to achieve the desired results by relying solely on either physics-based or keyframe animation.

1.3 Physics-Informed Neural Networks

Physics-informed neural networks (PINNs) are a class of machine learning algorithms that use physical knowledge in the form of partial differential equations (PDEs) and boundary conditions to train models which encode the underlying physical laws in those equations [3]. These algorithms have been used to fit physics equations to data and create models for a wide range of applications ranging from predicting arterial blood pressure [4] to predicting nanofluid viscosity [5]. PINNs can be trained using a common training method for neural networks in which a loss function and an optimizer are defined. Inputs are fed into the network and the network's output is evaluated using the loss function. The loss function evaluates the network's current predictions against its intended behavior, outputting a single value that measures how far off the predictions are. Larger values typically mean the neural network is not behaving in the desired manner and must be adjusted while smaller values mean that the network is performing well and needs less adjustment. During this process, automatic differentiation calculates the gradient of the PINN's loss function with respect to the weights in the neural network [6]. The gradient and loss function information is then passed into the optimizer, which is an algorithm that adjusts the weights of the neural network in a way that will minimize the value of the loss function. This process is then repeated in a loop until the loss measured by the loss function stagnates. At this point, the model is said to have converged into a solution. What distinguishes a PINN from other neural networks is that its loss function includes a term that involves solving differential equations using the output and gradient of the neural network itself. This makes it so

that as training progresses, the network and its gradients approximate a solution to those equations [7].

1.4 Objective of Research

In the context of computer animation, a PINN's boundary conditions can be treated as keyframes, allowing the animator to control the behavior of the simulation at specific points in time. By using boundary conditions in this way and leveraging the PINN's reliance on partial differential equations to interpolate between them, the animator will be able to meet the specific needs of the animation while using fewer keyframes and obtaining the accuracy and realism of a physics-based approach. This would provide the animator with a powerful tool that incorporates the benefits of both keyframe and physics-based animation, thus improving their ability to create compelling animations while saving time and effort. This research investigates the effects of various parameters and types of boundary conditions when training PINNs to gain a deeper understanding of how they can effectively be used to create animations. Specifically, it focuses on exploring the impact of the following:

- Weights assigned to each term in the loss function: The PINN used in this paper is trained using a loss function with three terms. Two of these constrain the solution to match physics. The third constrains the solution to match the boundary conditions, which work similarly to keyframes in this context. The purpose of exploring the effects of these weights is two-fold. The first is to find a set of weights where the PINN can accurately interpolate in a physics-based manner. The second is to determine if adjusting these weights can control the degree to which the PINN interpolates in a completely physics-based manner vs interpolating in a purely keyframe or boundary condition-based way.

- Number of collocation points used during training: Collocation points refer to the locations where the PINN is trained to satisfy the physical system being modeled and are what is used to calculate the physics terms of the loss function. By varying the number of collocation points used when training the PINN, this research intends to investigate their impact on the accuracy and convergence of the PINN to find out approximately how many are needed to achieve a good-looking animation.
- Type of boundary condition: Boundary conditions can be specified for different aspects of the physical system that the PINN is approximating. This parameter was investigated to test whether it was more effective to place boundary conditions on the outputs of the network, its gradient, or both. In the PINNs used in this research, the output of the network represents the position of the animated object, and the gradient of the network represents its velocity.
- Using a single boundary condition (start only, end only).
- Using a variable number of boundary conditions linearly spaced across the time of simulation, including conditions at the start and end time: By trying different strategies for placing boundary conditions, this research hopes to identify any strategy that leads to interpolation that has any unique characteristics or looks significantly better than another. Additionally, it seeks to identify approximately how many boundary conditions are needed to achieve acceptable interpolation along the desired trajectory. This measurement can be compared to the number of keyframes required to achieve an animation using traditional keyframe animation methods. This can be used to determine whether animating with a PINN can reduce the number of keyframes that need to be specified.

- Physical realism of boundary conditions: It can be difficult for animators to place keyframes so that an object is animated exactly according to physics. Additionally, animators often want to animate objects with a physics-inspired motion but do not seek to match physics exactly. It is common practice to exaggerate certain parts of an animation to make it more aesthetically pleasing. Because of these reasons, it is expected that animators will often choose boundary conditions in a way that physics cannot possibly satisfy them all. By exploring the limits of the PINN's ability to model increasingly non-realistic boundary conditions, this research aims to provide insight into the trade-off between accuracy and artistic expression when using PINNs for interpolation.

To achieve this goal, this research will train various PINNs that simulate the motion of an object attached to a two-dimensional Hookean spring while systematically varying parameters and boundary conditions. The results of this experiment will provide insight into the optimal use of PINNs for animation and could lead to new approaches for creating visually appealing animations.

The rest of this paper is organized as follows. Section 2.1 details the architecture of the PINN used in this paper. Section 2.2 explains the implementation of the PINN's loss function. Section 2.3 describes the process used to explore the effects of different hyperparameters and types of boundary conditions on training the network. Section 3 explores the results of the data generated from the process detailed in section 2.2. Section 4 discusses how to efficiently pick boundary conditions and parameters when animating with a PINN, details the advantages and limitations of this method, and details possible future work.

2. METHODS

2.1 Network Architecture

The PINNs for this research were defined to model the trajectory of an object attached to a two-dimensional Hookean spring. Given an input time, t, the network, f, was trained to predict the x and y coordinates of a point mass attached to a spring at that time. An illustration of this system is provided in Figure 2.

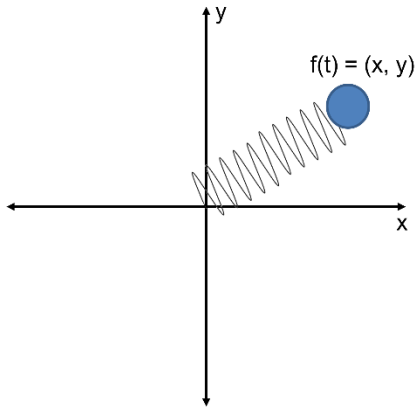

Figure 2: An illustration of the two-dimensional spring system modeled by the PINN. It consists of a spring with one free end, and one end fixed to the origin. A point mass is attached to the free end. Given an input time of t, the PINN was trained to predict the x and y coordinates of the point mass attached to the end of a spring at that time.

The network consisted of a linear input layer (1x16), four linear hidden layers (16x16), and a linear output layer (16x2) and used a tanh activation function. This structure is demonstrated in Figure 3.

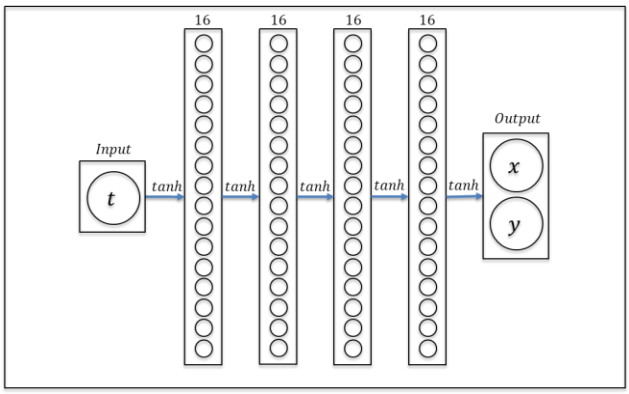

Figure 3: Architecture of the trained PINNs.

2.2 Loss Function

The loss function used to train the PINNs consisted of three terms. To constrain the solution to satisfy the physics of the Hookean spring, two physics-based loss terms were applied to the model. The first was derived from the conservation of momentum, and the second from the conservation of energy. To constrain the solution to satisfy the boundary conditions specified at keyframes by the animator, a third loss term measuring how closely the neural network met those conditions was used. The overall loss function was:

$$L_{total} = w_m L_m + w_e L_e + w_b L_b \,, \tag{1}$$

where L_m, L_e, and L_b are the overall loss derived from conservation of momentum, conservation of energy, and boundary conditions respectively and w_m, w_e, and w_b are numerical values that specify the relative influence of each loss term.

2.2.1 Conservation of Momentum

To calculate this term, the force acting on the object whose motion is being approximated must be known. For a Hookean spring, this is:

$$F_{spring} = kx, \tag{2}$$

where k represents the spring constant and x represents the offset of the spring from its resting position. Newton's second law can then be applied to set the force applied by the spring equal to the mass of the attached object, m, times its acceleration, a. The equation then becomes:

$$kx = ma. \tag{3}$$

Because the resting position of the spring is located at the origin in the system modeled by the PINN, the predicted position of the point mass output by the network and the offset of the spring from its resting position are the same. Thus x can be directly substituted with the output of the network, f. Likewise, because acceleration is the second derivative of position with respect to time, and the input of the PINN represents time, it can be substituted with the second derivative of the network with respect to its input. After these substitutions, the final form of the equation becomes:

$$kf = m\frac{\partial^2 f}{\partial t^2}. \tag{4}$$

Because this condition must be met in a physically conforming system, it is possible to take the difference between the left and right sides of the equation to get a measure of the loss at a single collocation point. To quantify the overall loss for the conservation of momentum term, the differences across all collocation points were aggregated by taking the mean square error:

$$L_m = \frac{1}{n}\sum_{1}^{n}(kf - m\frac{\partial^2 f}{\partial t^2})^2. \tag{5}$$

2.2.2 Conservation of Energy

To calculate this term, the total amount of energy in the system must be known. This can either be specified by the user or calculated from a boundary condition where position and velocity are both specified. For a two-dimensional spring system, potential energy and kinetic energy can be calculated using:

$$PE = \frac{1}{2}kx^2, \tag{6}$$

$$KE = \frac{1}{2}mv^2. \tag{7}$$

The sum of these two values:

$$H_{total} = PE + KE, \tag{8}$$

was treated as the total amount of energy in the system. The energy at each collocation point used to train the PINN, H_{point}, was calculated by performing a substitution process similar to the one used to calculate the conservation of momentum term. Specifically, the output of the PINN, f, was substituted for x. Because velocity is the first derivative of position with respect to time, the first derivative of the output of the PINN with respect to its input was also substituted for v. These substitutions resulted in:

$$PE_{point} = \frac{1}{2}kf^2, \tag{9}$$

$$KE_{point} = \frac{1}{2}m\,(\frac{\partial f}{\partial t})^2, \tag{10}$$

$$\text{and } H_{point} = PE_{point} + KE_{point}. \tag{11}$$

Because of the principle of conservation of energy, the total energy in the system should stay constant. Thus, it is possible to quantify the loss at a single collocation point by taking the difference between the total energy in the system and the amount of energy calculated at that point. To obtain a measure of the overall loss for the conservation of energy term, the differences across all collocation points were aggregated using mean square error:

$$L_e = \frac{1}{n}\sum_{1}^{n}\left(H_{total} - H_{point}\right)^2. \tag{12}$$

2.2.3 Keyframes (Boundary Conditions)

When training the PINN, boundary conditions for position and velocity could be specified. A boundary condition consisted of specifying the exact position or velocity the PINN should output for a specific input time. The loss at a single position boundary condition was calculated by taking the difference between the PINN's predicted position of the object, f, and the desired position of the object at that time, $x_{boundary}$. Similarly, the loss at a single velocity boundary condition was calculated by taking the difference between the PINN's predicted velocity of the object, $\frac{\partial f}{\partial t}$, and the desired velocity of the object at that time, $v_{boundary}$. To obtain measures of the combined loss from all position and velocity boundary conditions, the position and velocity loss at every boundary condition was aggregated using mean square error. The aggregated values were then summed to get a final measure of the overall loss from boundary conditions:

$$L_b = \frac{1}{n}\sum_{1}^{n}\left(x_{boundary} - f\right)^2 + \frac{1}{n}\sum_{1}^{n}\left(v_{boundary} - \frac{\partial f}{\partial t}\right)^2. \tag{13}$$

2.3 Parameter and Boundary Condition Exploration

The result of training a PINN is affected by a variety of parameters. To better understand how to best use a PINN for interpolation and the limits of the method, it was decided to investigate the effects of varying the following easily tunable parameters: the weights assigned to each term of the loss function, the number of collocation points, the type of boundary condition, the placement of boundary conditions, the number of boundary conditions, and the physical realism of the boundary conditions.

2.3.1 Identifying Optimal Loss Function Term Weights

To ensure that the Physics-Informed Neural Network (PINN) was interpolating the spring's motion in a physics-based manner, a grid search was carried out. During this search, different combinations of weights for each loss term (w_m, w_e, and w_b) were used to train each PINN, with every combination of the values 1e-3, 1e-1, 0, 1e1, and 1e2 tested. For each iteration of the search, a trajectory generated by simulating the spring's motion was compared to the trajectory generated by a PINN trained with the initial position and velocity from the simulation as the only boundary conditions. The best-performing weights were considered to be those that produced PINNs whose predicted trajectory most closely matched the trajectory from the simulation. During this search, all parameters involved in training the PINN aside from the loss term weights remained constant. Appendix A contains the exact configuration of parameters used for both the simulation and the PINN throughout this process.

2.3.2 Exploring the Effects of Other Parameters

After identifying the best-performing set of loss term weights for the PINN, a subsequent grid search was carried out to explore the impact of different boundary conditions setups and other parameters on the PINN's performance. Specifically, the number of collocation points, the type of boundary condition, the placement of boundary conditions, the number of boundary conditions, and the physical realism of the boundary conditions were varied. Every combination of the following values for the parameters was tried:

- Number of collocation points: 1, 10, 100. These were spaced out linearly across time for the duration of the animation.
- Type of boundary conditions: Position only, velocity only, both position and velocity together.

- Placement of boundary conditions: One boundary condition at the start of the trajectory, one boundary condition at the end of the trajectory, a variable number (2, 3, 7, 15, 31) of boundary conditions linearly spaced in time including the start and end conditions.
- Physical realism of the boundary conditions: guaranteed to be on a physical path, approximating a physical path, completely nonsensical.

The mass and spring constant were set as learnable parameters with an initial value of 100. All other parameters including the weights for the loss terms were held constant. A complete list of the parameters used for this search can be found in Appendix B.

3. RESULTS

3.1 Loss Term Weights

After testing various combinations of weights for each term of the loss function, the weights which caused the PINN's approximation of the trajectory to match the simulation's trajectory most closely were the ones that minimized the energy conservation term (w_e) and prioritized the boundary condition term (w_b) over the conservation of momentum term (w_m). With careful selection of weights, it is possible for the PINN to practically match the behavior of the simulation even when given minimal information of only the starting position and velocity as boundary conditions. This is demonstrated in Figure 4.

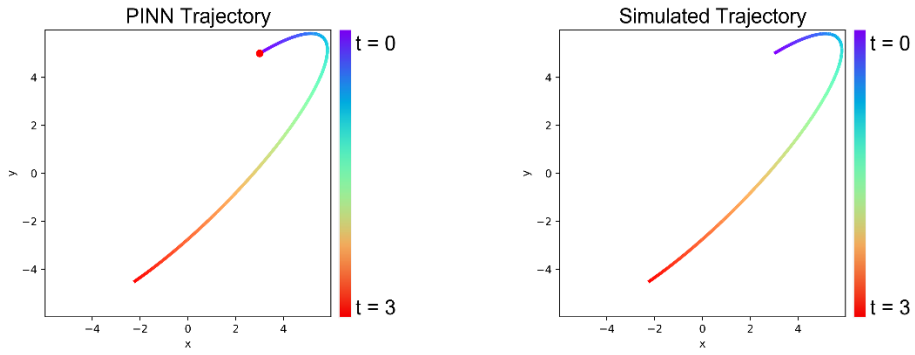

Figure 4: Two plots demonstrating the trajectory of an object attached to a spring. The x and y positions of the mass attached to the spring are parameterized in time. The plot on the right shows the trajectory predicted by a simulation. The plot on the left shows the trajectory by a PINN trained with only the starting position and velocity of the simulation as boundary conditions. The position boundary condition is denoted by a red dot.

Experimenting with different sets of weights also revealed that it is crucial to select these carefully because it is possible to end up with weights that do not perform interpolation in any useful way. Examples of this behavior are demonstrated in Figure 5.

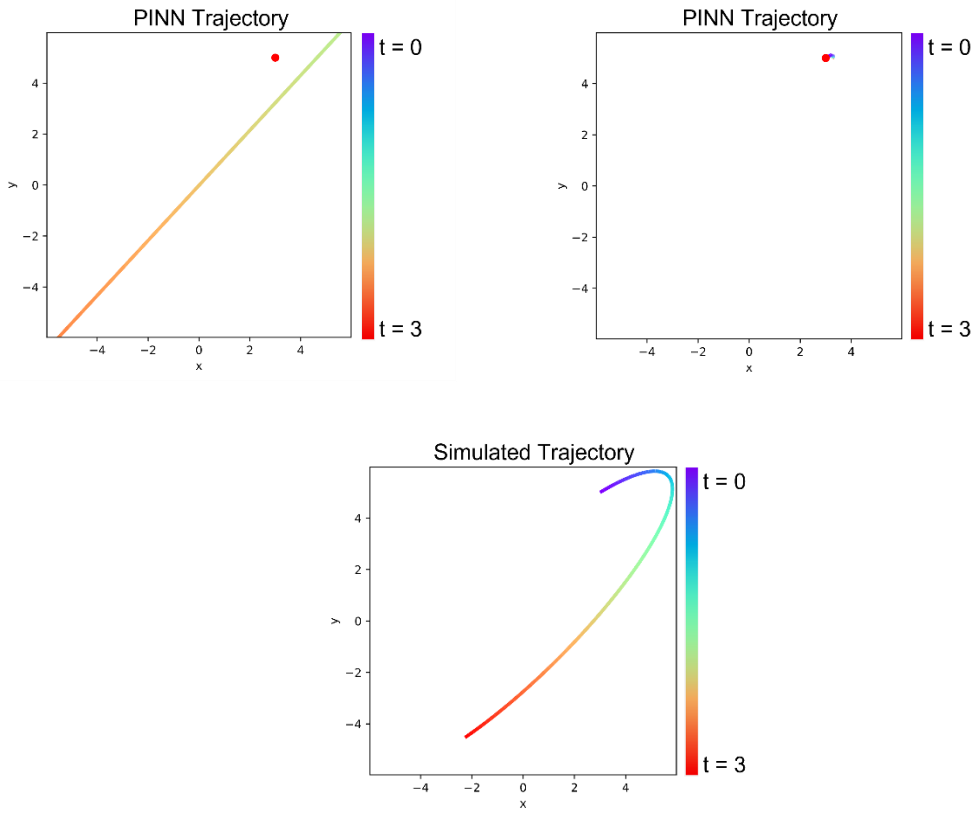

Figure 5: Three plots demonstrating the trajectory of an object attached to a spring. The x and y positions of the mass attached to the spring are parameterized in time. The plots on the top show the trajectories predicted by PINNs trained with weights that did not work well and using only the starting position and velocity of the simulation as boundary conditions. The position boundary condition is denoted by a red dot. The plot on the bottom shows the trajectory predicted by the simulation and is what the top two plots should ideally look like.

Regarding controlling the degree of accuracy vs artistic intent in the PINN's interpolation using these weights, it does not seem to be very promising. The weights will either cause the PINN to converge to a mostly physically accurate trajectory or an unusable one.

3.2 Number of Collocation Points

Regarding the number of collocation points used for training PINNs, it was found that using 10 points is the minimum required for physics-based keyframe interpolation. However, using such a small number of points occasionally resulted in trajectories with segments that were not very physically believable and could look awkward. Increasing the number of collocation

points to 100 improved the accuracy of the trajectories and consistently produced PINNs that closely followed the physics laws they were trained on. However, this also led to an increase in training time, from two and a half minutes for 10 points to roughly 28 minutes for 100 points. This increased training time may not be ideal for animation workflows where being able to iterate quickly is desired. Therefore, a balance between physical accuracy and training time should be considered when training.

3.3 Number of Boundary Conditions

The number of boundary conditions has some impact on the resulting trajectory predicted by the trained PINN. At least one boundary condition is always required because the PINN will fail to converge to a solution if none are present. As the purpose of the PINN used in this paper was to do keyframe animation with physics-based interpolation, and the boundary conditions are the keyframes in this context, the best results should show the PINN's predicted trajectory of the animated object generated passing through each boundary position as closely as possible. This was achieved consistently when the PINN only had to account for smaller numbers of boundary conditions. However, as the number of boundary conditions increased, the PINN sometimes became unable to meet all of them and would instead converge to a trajectory that minimized the boundary condition loss term by "averaging out" boundary conditions in a way. This behavior is displayed in Figures 6 and 7.

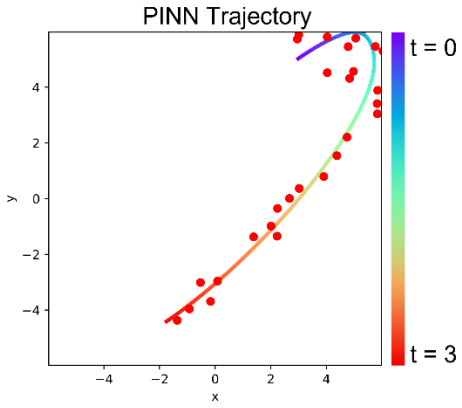

Figure 6: A plot demonstrating the trajectory of an object attached to a spring. The x and y positions of the mass attached to the spring are parameterized in time. The positions of boundary conditions are denoted by red dots and are placed roughly on a physically possible path. The plot does not necessarily specifically meet any boundary condition but rather appears as a type of "best fit" line.

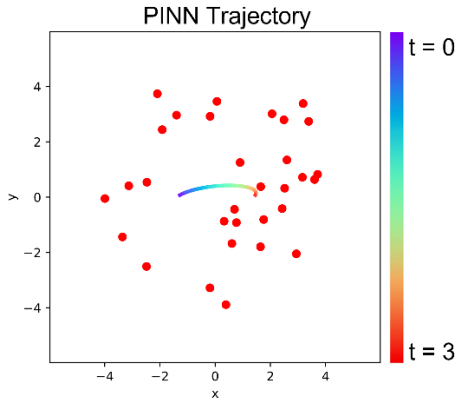

Figure 7: A plot demonstrating the trajectory of an object attached to a spring. The x and y positions of the mass attached to the spring are parameterized in time. The positions of boundary conditions are denoted by red dots and are all placed on a completely nonsensical path.

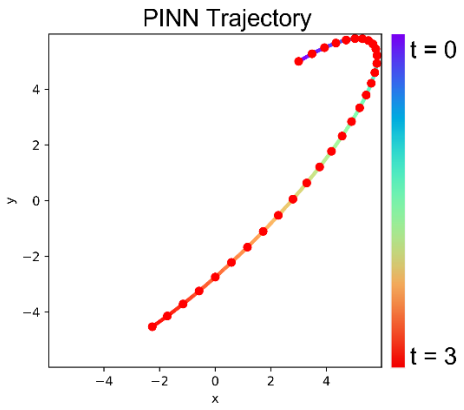

Figure 8: A plot demonstrating the trajectory of an object attached to a spring. The x and y positions of the mass attached to the spring are parameterized in time. The positions of boundary conditions are denoted by red dots and are all placed on a physically possible path.

However, as shown in Figure 8, in the case where the boundary conditions were all located on a physically possible path, the PINN was able to produce good results for keyframe interpolation. This suggests that increasing the number of boundary conditions in and of itself is not what is primarily responsible for hindering the PINN's ability to successfully interpolate between keyframes. This behavior is rather caused by the placement of boundary conditions because in all the cases where the PINN was unable to meet all the boundary conditions, there was no physically possible trajectory connecting the boundary conditions. Increasing the number of boundary conditions not located on a physically possible trajectory simply increased constraints on the PINN and the likelihood that it would be impossible to meet them all while still interpolating in a physics-based manner.

3.4 Placement of Boundary Conditions

The placement of boundary conditions has a large impact on the resulting animation. When boundary conditions were placed on a trajectory that was physically possible, the PINNs were consistently able to find a solution that closely met every boundary condition. When boundary conditions were placed on a path that was physics-inspired but not necessarily

possible, the PINNs converged to a solution that approximated it but did not necessarily hit any of the boundary conditions as shown above in Figure 6. When boundary conditions were placed on a completely nonsensical path, the network often converged to a solution that still appeared to follow a physically based trajectory. However, the resulting trajectory didn't come anywhere near the boundary conditions, nor did it make any sense. This is demonstrated above in Figure 7.

3.5 Type of Boundary Conditions

On its own, changing the type of boundary condition used to train the PINN did not seem to have any significant effects. If the boundary conditions were at least approximately physically possible, similar trajectories were produced regardless of boundary condition type. This is shown in Figure 9.

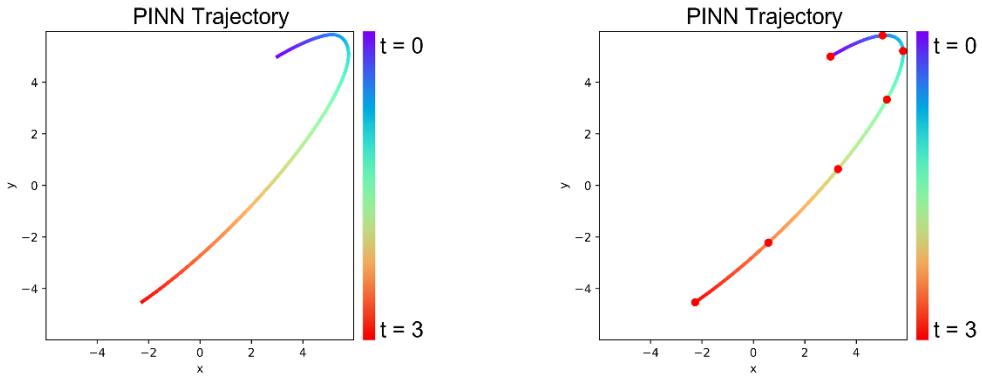

Figure 9: Two plots demonstrating the trajectory of an object attached to a spring. The x and y positions of the mass attached to the spring are parameterized in time. The plot on the left shows the trajectory predicted by a PINN trained with only 7 velocity boundary conditions linearly spaced along time. The plot on the right shows the trajectory predicted by a PINN with only 7 position boundary conditions linearly spaced in time. The position boundary conditions are denoted by red dots.

When the boundary conditions were not physically possible, using different types of boundary conditions while holding other variables constant also did not lead to any PINNs that were more usable than the others.

4. CONCLUSION

4.1 Choosing Optimal Parameters and Boundary Conditions

Careful selection of parameters and boundary conditions is crucial for achieving a PINN that converges to a solution that can be used to produce a visually appealing animation. This section discusses what were observed to be the best ways to set each parameter.

4.1.1 Loss term weights

The results of the investigation reveal that the loss term weights are one of, if not the most important parameters that impact the results from the PINN. Without setting these properly, the PINN will fail to converge to a useful solution. The best weights give the most importance to the boundary condition, conservation of momentum, and conservation of energy terms in that order. During the initial gird search, values of 1e-1, 1e-3, and 0 for w_b, w_m, and w_e were found to be particularly effective at ensuring the PINN could accurately interpolate in a physics-based manner. With these weights, it was possible to get a near-exact match of the simulated trajectory using 100 collocation points and the initial position and velocity as the only boundary conditions. This was demonstrated in Figure 4. Choosing weights similar to these is thus strongly recommended.

4.1.2 Boundary Conditions

Aside from the weights of each term in the loss function, the physical plausibility of the boundary conditions has the most influence on the results from the PINN. Using boundary conditions for which it was possible for a physical path to exist or closely approximate resulted in PINNs with that very closely met those boundary conditions and were thus well suited for physics-based keyframe interpolation. One easy method of ensuring that boundary conditions

maximize the chances that the boundary conditions will all be located on a physically possible path is to only use a few positions as boundary conditions. This reduces the chances that the boundary conditions will contradict each other and avoids over-constraining the PINN, thus enhancing its ability to find a useful solution. By using one to three positions as boundary conditions, the PINNs trained throughout the course of this research were able to reliably model a trajectory that passes through them.

The type of boundary condition (position or velocity) used during training does not seem to affect the results in any significant manner. That said, when animating trajectories, it is recommended to primarily use only position boundary conditions because their effects are more easily visualized. Setting a velocity boundary condition at a single point in time may be useful and help direct the network toward a desired trajectory. However, specifying both velocity and position conditions at every point in time for which a boundary condition is specified can make it easier to feed contradicting and/or physically impossible information into the PINN. This can lead to awkward trajectories for the animation, even when only specifying boundary conditions for a small number of points in time.

4.1.3 Collocation Points

The number of collocation points is also an important parameter to take into consideration. Using 100 collocation points resulted in the best performance, with the modeled trajectories most closely following physical laws. However, visually comparable results can still be achieved with as little as 10 collocation points. That said the trajectory from using a smaller number of points will be less physically accurate. Because it takes considerably longer to train PINNs as the number of collocation points increases, a recommended workflow for an animator could be to use fewer collocation points to quickly train a “preview” PINN and get a general idea

of how the PINN will interpolate the boundary conditions. The animator can then adjust the boundary conditions as needed and train the PINN again until they have an animation they are fairly satisfied with. At this point, they can then train a final PINN using more collocation points to improve the accuracy of the physics-based interpolation.

4.2 Advantages of Method

This research demonstrates the abilities of PINNs to interpolate physics-based motion very accurately with as little information as a single boundary condition. Additionally, the results of this research suggest using PINNs for keyframe interpolation has potential. When it is physically possible for all the boundary conditions to be met, a PINN treats them like keyframes and produces animations that reliably position the object as desired at specific times. This is demonstrated in Figure 10.

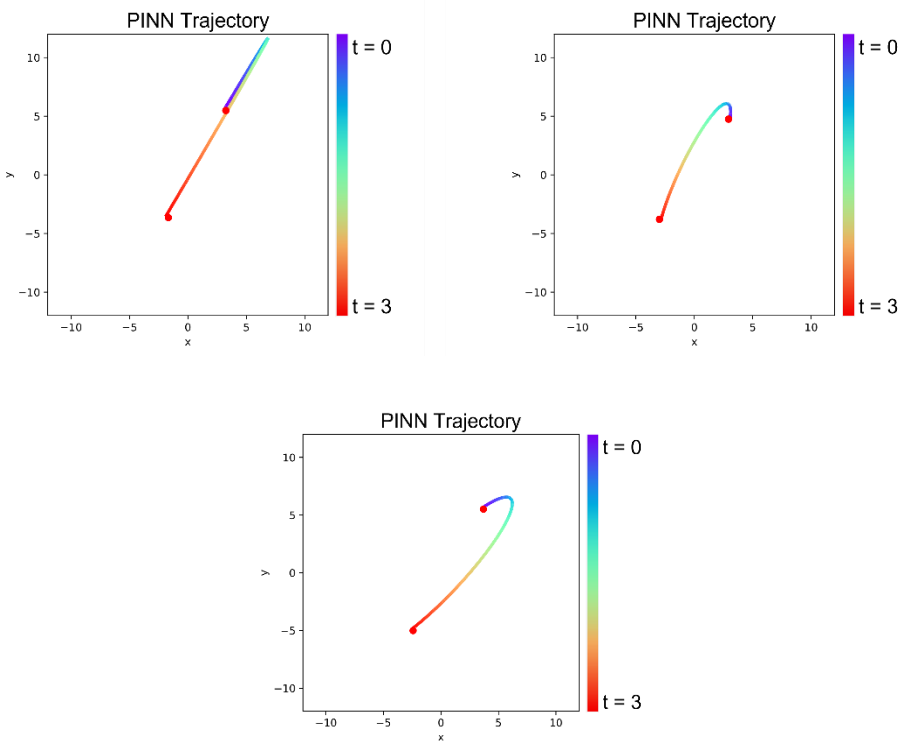

Figure 10: Plots demonstrating the trajectory of an object attached to a spring predicted by PINNs trained using only start and positions as boundary conditions. The x and y positions of the mass attached to the spring are

parameterized in time. The positions of boundary conditions are denoted by red dots. Each plot shows the trajectory generated using a different PINN and set of boundary conditions.

However, because the interpolation is physics-based, fewer keyframes are needed to achieve the animation than would be needed using traditional keyframe interpolation methods. This could allow the artist to put the time that would've been spent keyframing into improving other aspects of the animation.

In situations where the PINN can't meet all the boundary conditions, but the conditions are still roughly on a physically possible path, the PINN will interpolate the boundary conditions so that they are all met as closely as possible. In this case, the trajectory from the PINN resembles more of a line of best fit rather than a line that passes through each boundary condition. This behavior is shown in Figure 6. Although this is not optimal for keyframe animation because the animated object is no longer guaranteed to be positioned as desired at specific times, this property of PINNs could be useful if applied to animation techniques involving real-world measurements, which are often noisy. The measurements from these systems can be used as the boundary conditions in this context, and the PINN can find a physics-based trajectory that fits the data well.

4.3 Limitations of Method

Although using a PINN to interpolate between keyframes has the potential to be an effective tool to create physically based animations, it has some significant limitations that need to be considered. One of these is that the animator must understand the physical systems involved in their animation, and differentiable equations that describe the physical constraints of the system must exist. Without these, it is not possible to create a PINN capable of representing the motion of that physical system. Another limitation of using PINNs is that they require a careful selection of parameters and boundary conditions during training. If these are chosen

poorly, a PINN is highly likely to fail to converge to a solution that produces quality animations. Moreover, the process of training a PINN and searching for optimal parameters can be time-consuming and computationally expensive, which can limit an animator's ability to quickly iterate on their work and restrict the scenarios in which this method of interpolation can be applied. Because of this, it may be more efficient to use traditional methods to animate certain types of physics-based motion.

4.4 Future Work

Although the PINNs examined in this paper are limited in scope to the example of a single particle attached to a two-dimensional spring, animating with PINNs has shown some interesting results and there is potential to develop this concept further in various ways.

4.4.1 Modelling the motion of a rope

One way to extend this work would be to use PINNs to animate a rope that is continuous in space as opposed to a single particle. To do this, the architecture of the PINN in this paper could be modified so that it takes in a second input parameter, u. The modified neural network architecture is shown in Figure 11.

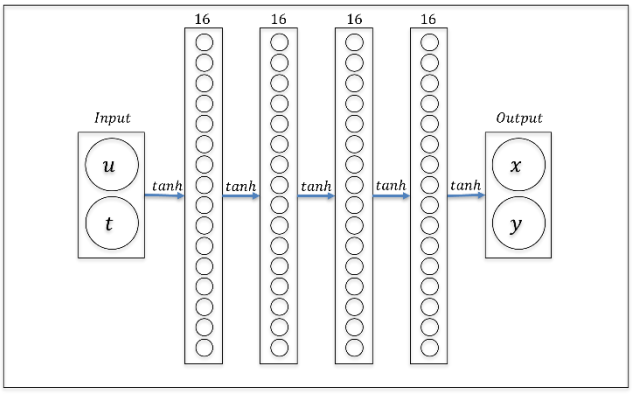

Figure 11: Architecture of the PINN to model a continuous rope.

This u parameter represents the location of the point we want to predict along the length of the rope. For example, if we have a rope of unit length, u will range from 0 at one end to 1 at the other end. The point in the middle of the rope would correspond to a u value of 0.5.

To account for the forces acting on the rope, the physics term of the loss function would also need to be modified. One approach could be to define the elastic potential of the rope as:

$$V = k\left(\frac{\left|\frac{\partial x}{\partial u}\right|}{\left(\frac{\partial x}{\partial u}\right)_0} - 1\right)^2, \tag{14}$$

where x represents the position of a point in the rope, k represents the spring constant, and $\left(\frac{\partial x}{\partial u}\right)_0$ represents the gradient of x with respect to u at time 0. The forces acting on the rope, could be defined to be the force of gravity and the negative derivative of the elastic potential with respect to x:

$$F = -\frac{\partial V}{\partial x} + mg, \tag{15}$$

where m is the mass of the rope. Newton's second law could then be applied to set the forces acting on the rope equal to its mass times its acceleration, a:

$$ma = -\frac{\partial V}{\partial x} + mg. \tag{16}$$

To make the model continuous across space, it can be converted to an infinitesimal version by dividing both sides of the equation by the volume of the rope. This results in:

$$\rho a = -\frac{\partial V}{\partial x} + \rho g, \tag{17}$$

where ρ is the density of the rope. To apply these equations to the PINN, the physics variables can be replaced with the corresponding outputs and gradients of the network. The position of a point in the rope, x, is represented by the output of the network, f, and can thus be substituted.

Likewise, the acceleration, a, can be replaced by the second derivative of the network with respect to time. The force equation thus becomes:

$$\rho \frac{\partial^2 f}{\partial t^2} = -\frac{\partial V}{\partial f} + \rho g, \tag{18}$$

and V becomes:

$$V = k(\frac{\left|\frac{\partial f}{\partial u}\right|}{(\frac{\partial f}{\partial u})_0} - 1)^2. \tag{19}$$

Because conservation of momentum requires both sides of the force equation to be equal, the difference between the left and right side of the equation can be used as a measure for the physics loss at a single collocation point. Thus, the physics loss at a single collocation point is:

$$L = \rho \frac{\partial^2 f}{\partial t^2} + \frac{\partial V}{\partial f} - \rho g. \tag{20}$$

The loss at each collocation point can then be aggregated into a single value and then combined with the loss from boundary conditions. These modifications should be sufficient to enable the network to simulate the motion of a rope which is continuous in space.

4.4.2 Other directions

Future directions for this work could also include using PINNs to interpolate different types of motion. For example, a PINN could be used to interpolate the motion of rotating objects. Another example could be interpolating the motion of multiple unconnected objects that influence each other's trajectories due to forces such as gravity. PINNs could also potentially be used to interpolate motion that involves collisions.

Another future direction for this work would be to experiment with different ways of modifying the loss function so that the PINN places more emphasis on meeting boundary conditions rather than ensuring physical accuracy. A basic expectation for keyframe interpolation is that the animation hits all the keyframes. However, the PINNs in this paper do not consistently

meet this expectation because they do not interpolate through boundary conditions placed on sufficiently impossible trajectories. This severely limits their use for this purpose. Addressing this issue would greatly expand the range of applications for using PINNs to interpolate keyframes.

Alternatively, future work could explore the effects of using different types of boundary conditions for the training of PINNs. The boundary conditions of the PINNs studied in this research consisted of having the PINN match exact positions and/or velocities at specific points in time. However, it would be interesting to investigate the impact of different types of boundary conditions on the training process such as direction-based constraints. These could be more intuitive for animators to set, as they are often concerned with the direction of motion rather than an exact velocity. Additionally, instead of checking for a precise position or velocity at a given time, range-based constraints could be utilized. For example, the position of the animated object could be constrained to always remain within a certain distance from a specific point, or the velocity of the object could be constrained so that its magnitude is always within certain bounds. Experimenting with different types of boundary conditions could lead to more intuitive or adaptable PINNS for animation.

Another future direction would be investigating how to best implement PINNs into an animator's toolkit and comparing the process of making physics-based animations with a PINN to that of traditional methods. This could involve developing user-friendly interfaces and tools for animators to use PINNs in their workflow. The comparison could involve exploring the creative limitations and possibilities of using PINNs for interpolation and evaluating the speed and effectiveness of the method relative to traditional animation techniques when creating different types of animations.

REFERENCES

[1] A. Chopine, ‘Chapter 8 - Animation: It’s Alive!’, in 3D Art Essentials, A. Chopine, Ed. Boston: Focal Press, 2011, pp. 103–126.

[2] D. H. House and J. C. Keyser, *Foundations of physically based modeling and Animation*. Boca Raton, Florida: CRC PRESS, 2016.

[3] M. Raissi, P. Perdikaris, and G. E. Karniadakis, ‘Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations’. arXiv, 2017.

[4] G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, and P. Perdikaris, ‘Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks’, Computer Methods in Applied Mechanics and Engineering, vol. 358, p. 112623, 2020.

[5] M. Islam, M. S. H. Thakur, S. Mojumder, and M. N. Hasan, ‘Extraction of material properties through multi-fidelity deep learning from molecular dynamics simulation’, Computational Materials Science, vol. 188, p. 110187, 2021.

[6] “Optimizing model parameters,” *Optimizing Model Parameters - PyTorch Tutorials 1.13.1+cu117 documentation*. [Online]. Available: https://pytorch.org/tutorials/beginner/basics/optimization_tutorial.html.

[7] S. Cuomo, V. S. di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, ‘Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What’s next’. arXiv, 2022.

APPENDIX: A

Network Parameters that remained constant during the grid search for loss function term weights:

- Number of collocation points (100)
- Boundary conditions:
    - Position: Spring is at an initial displacement of (3 m, 5 m) at the start of the simulation (t = 0)
    - Velocity: The object is traveling with a speed of (5 m/s, 3 m/s) at the start of the simulation (t = 0)
- Network architecture (activation function, number of layers, size of layers, etc.)
- Learning rate (1e-4)
- Optimizer (Adam)
- Maximum number of epochs (20,000)
- Duration of simulation (3 seconds)
- Mass of object attached to spring (100 kg)
- Spring constant (100 N/m)
- Resting length of spring (0 m)

APPENDIX: B

Network Parameters that remained constant during the grid search to explore the effects of other parameters:

- Network architecture (activation function, number of layers, size of layers, etc.)
- Learning rate (1e-4)
- Optimizer (Adam)
- Maximum number of epochs (50,000)
- Duration of simulation (3 seconds)
- Initial mass of object attached to spring (100 kg)
- Initial spring constant (100 N/m)
- Resting length of spring (0 m)