75th International Astronautical Congress (IAC), Milan, Italy, 14-18 October 2024.

IAC-24-C1.9.8

Generation and Classification of Critical Points in Uncertain N-Body Problems via Machine Learning

Callum Wilson[1, a)*] **and Massimiliano Vasile**[1]

[1] *Department of Mechanical and Aerospace Engineering, University of Strathclyde, 75 Montrose Street, Glasgow G1 1XJ, United Kingdom*
[a] *callum.j.wilson@strath.ac.uk*
[*] *Corresponding author*

Abstract

The study of dynamical systems is relevant for a wide variety of problems in astrodynamics. These are systems defined by differential equations that may have no known analytical solution. Despite this, the study of such systems gives rise to a rich variety of phenomena that are interesting in both a theoretical sense and for practical applications.

One such example is the location of critical points. In astrodynamics, these are commonly studied in the context of equilibrium points in the N-body problem. Equilibrium solutions can be found numerically for some restricted cases of the full system. Machine learning techniques are frequently used to find patterns from data and are therefore a promising technique for studying patterns in dynamical systems. In particular, recent advances in generative models have shown remarkable performance in synthesizing new data based on examples.

This work demonstrates the application of physics-informed generative models for learning distributions of critical points in astrodynamics. The class of generative models used are flow-based models, which transform a tractable distribution into a more complex one that allows direct sampling of new points. The architecture proposed in this paper integrates the generative layers with a classification of the critical points according to their stability properties. In many dynamical systems of interest, in fact, equilibrium points may be characterised as stable, unstable, or metastable depending on the system's behaviour in the region of the critical point. In addition to generating new points, the proposed method classifies the critical points by learning in a semi-supervised manner from the classes of known critical points. This allows further analysis of how certain types of solution are distributed in the system. The methodology is applied to equilibrium solutions in the N-body problem with uncertain parameters.

Keywords: N-body Problem, Generative AI

1 Introduction

Machine Learning (ML) methods are now often applied to problems that are difficult to solve using conventional approaches. One of the main features of these methods is their ability to extract patterns in data. Most early research used this to solve classification tasks by finding similarities between objects of the same class. Now there are new generative architectures that can produce new data with patterns similar to previously seen data [1]. This popular new field of generative models has seen mainstream usage in text, image, and video generation [2, 3].

One area that shows potential to benefit from ML is the study of dynamical systems. These mathematical models are applied to a wide variety of real-world systems to explain different phenomena from vibrating bodies to the motions of planets. In such models we are often interested in their critical points that can represent equilibrium points or periodic solutions of the system [4]. Traditional approaches to finding these points use numerical methods to find roots or minima of some mapping representing the system dynamics. In systems modelling orbital dynamics, this mapping is typically related to the energy of the system [5].

This work investigates the potential for using generative models to determine critical points of dynamical systems. This problem is formulated as modelling a probability distribution that has a high density at the location of critical points. To model the distribution, we use a class of generative models referred to as flow-based models, which map a complicated probability distribution to a more tractable distribution. We consider the problem of finding equilibrium points in the Circular Restricted Five-Body Problem (CR5BP) [6].

Other applications of ML methods to dynamical systems have focussed on creating reduced order models of dynamical systems, both through deep learning and simpler ML systems [7]. These models are useful for prop-

agating complex systems or deriving controllers for such systems. For example, in [8] the authors create a latent embedding of image trajectories using variational autoencoders such that these trajectories are locally linear. This allows the use of conventional optimal control methods for solving complex non-linear problems. The methods proposed here focus more specifically on finding solutions to dynamical systems without directly modelling the transient behaviour of the system.

An earlier related work used Particle Swarm Optimisation (PSO) to find periodic orbits in a dynamical system [9]. The authors demonstrate that PSO is capable of finding solutions for systems that are otherwise difficult to solve with traditional root-finding algorithms. The added benefit of using a flow-based model instead of a swarm approach is the representation of the distribution of solutions in a latent space that allows efficient sampling.

The remainder of the paper is organised as follows. Section 2 describes the proposed method of using flow-based model used to sample critical points. Section 3 then goes on to describe the formulation of the CR5BP to which we apply the model and the generation of data for training. Results of training several model configurations are presented in section 4 and section 5 gives conclusions and future work.

2 Methods

2.1 Flow-based Generative Model

This class of generative models, also referred to as "Normalising Flows", transform a prior distribution $p_{\mathbf{Z}}(\mathbf{z})$ through a series of invertible transformations to a target distribution $p_{\mathbf{X}}(\mathbf{x})$ [10]. The random variable $\mathbf{Z} \in \mathbb{R}^D$ has a known and tractable distribution and the transformation is defined as $\mathbf{Z} = f(\mathbf{X})$. The change of variable rule gives the following expression for the probability density of $\mathbf{X}$:

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(f(\mathbf{x}))\det\left(|\mathrm{D}f(\mathbf{x})|\right) \tag{1}$$

where $\mathrm{D}f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian of f at $\mathbf{x}$. To generate a sample $\mathbf{x}$ from $\mathbf{X}$, one can first sample from the simpler distribution of $\mathbf{Z}$ and then transform this sample via $\mathbf{x} = f^{-1}(\mathbf{z})$. This is the main benefit of flow-based models, where the transformation f represents a "flow" from a potentially intractable distribution to a much simpler one.

There are several different architectures of flow-based generative models [11]. The architecture used here is called Non-linear Independent Components Estimation (NICE) [12], which is a volume-preserving flow since the prior distribution has the same dimensionality as the target, $f : \mathbb{R}^D \to \mathbb{R}^D$. The transformation f is a composition of several "layers" of transformations $f = f_L \circ \ldots \circ f_2 \circ f_1$. Each transformation is a bijective coupling layer with a tractable inverse and unit Jacobian determinant.

The output of layer f_l is denoted $h^{(l)}$, where $h^{(0)} = \mathbf{x}$. To make each layer invertible, the input is divided into two parts: h_{I1} and h_{I2} where the indexes I_1 and I_2 define how the vector is split. For the two-dimensional problems considered here, where $\mathbf{x} = (x, y)$, this is simply the first and second terms, i.e. $h^{(0)}_{I_1} = x$ and $h^{(0)}_{I_2} = y$. The additive coupling law is defined as

$$h^{(l+1)}_{I_1} = h^{(l)}_{I_2}, \tag{2}$$

$$h^{(l+1)}_{I_2} = h^{(l)}_{I_1} + m^{(l)}(h^{(l)}_{I_2}), \tag{3}$$

where m is a non-linear function defined for each coupling layer. This function is parameterised as a Deep Neural Network (DNN) such that each m is trainable. At least three coupling layers are required to allow all dimensions in the input to influence each other. As in the original work of NICE, the models trained here use $L = 4$ coupling layers.

Since each coupling layer is designed to have unit Jacobian determinant, these layers alone do not allow for scaling of the inputs. Therefore, an additional scaling layer is used in the output of the NICE model. This scaling layer has the form

$$\mathbf{z} = S \odot h^{(L)}, \tag{4}$$

where S is a diagonal scaling matrix with $S_{ii} = e^{s_i},\ i = 1, \ldots, D$. One other distinguishing feature of NICE is it uses a prior distribution that is factorial, i.e.

$$p_{\mathbf{Z}}(\mathbf{z}) = \prod_{i=1}^{D} p_{\mathbf{Z}_i}(\mathbf{z}_i). \tag{5}$$

Here a logistic distribution is used as the prior:

$$p_{\mathbf{Z}_i}(\mathbf{z}_i) = \frac{e^{-\mathbf{z}_i/\sigma}}{1 + e^{-\mathbf{z}_i/\sigma}}, \tag{6}$$

where the variances σ_i^2 are also trainable parameters of the model. Since this distribution is factorial, this results in the "NICE criterion" shown in (7) that defines the log-likelihood of the modelled distribution.

$$\log(p_{\mathbf{X}}(\mathbf{x})) = \sum_{i=1}^{D} \left[\log(p_{\mathbf{Z}_i}(f(\mathbf{x})_i)) + \log(|S_{ii}|)\right] \tag{7}$$

2.2 Conditional Flow-based Model

In the case of conditional generative modelling, we seek to model a probability conditioned on a variable $\mathbf{c} \in \mathbf{C}$. This variable can be a real valued vector, such as defining one or more parameters of the target distribution, or a categorical one-hot vector defining different classes of points in the distribution.

Now the target distribution has the form $p_{\mathbf{X}}(\mathbf{x}|\mathbf{c})$. In this formulation of conditional model, the prior distribution remains the same as in the general flow-based model and the transformation f includes the variable $\mathbf{c}$. Equation (1) can then be rewritten as

$$p_{\mathbf{X}}(\mathbf{x}|\mathbf{c}) = p_{\mathbf{Z}}(f(\mathbf{x},\mathbf{c}))\det\left(|\mathrm{D}f(\mathbf{x},\mathbf{c})|\right). \tag{8}$$

The process for including $\mathbf{c}$ in the transformation is similar to that of conditional GLOW [13]. At each coupling layer, the input to m is concatenated with $\mathbf{c}$ such that Eqs. 2 and 3 become

$$h_{I_1}^{(l+1)} = h_{I_2}^{(l)}, \tag{9}$$

$$h_{I_2}^{(l+1)} = h_{I_1}^{(l)} + m^{(l)}\left(\begin{bmatrix} h_{I_2}^{(l)} & \mathbf{c} \end{bmatrix}\right). \tag{10}$$

This maintains the unit Jacobian determinant of NICE to allow straightforward density estimation. The conditional NICE criterion then becomes

$$\log(p_{\mathbf{X}}(\mathbf{x}|\mathbf{c})) = \sum_{i=1}^{D}\left[\log(p_{\mathbf{Z}_i}(f(\mathbf{x},\mathbf{c})_i)) + \log(|S_{ii}|)\right] \tag{11}$$

2.3 Loss Functions

A general flow-based model aims to maximise the likelihood of training data $\mathbf{X}_{train} = \mathbf{x}_n$, $n = 1,\ldots,N$ in the model as defined by the NICE criterion in (7). This is equivalent to minimising the KL-divergence between model samples and true samples in physical space [11]. The objective is then defined as

$$\max_{\theta} \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}} \sum_{i=1}^{D}\left[\log(p_{\mathbf{Z}_i}(f(\mathbf{x})_i)) + \log(|S_{ii}|)\right], \tag{12}$$

where θ denotes the trainable parameters of the model.

One of the known issues with training flow-based models using only maximum likelihood is the quality of samples generated by a trained model [14]. Flow-GAN is a method for training flow-based models that combines the advantages of easy density estimation of flow-based models with the high quality sampling of Generative Adversarial Networks (GANs). In this formulation, the generator $\mathcal{G}_\theta(\mathbf{z}) = f^{-1}(\mathbf{z})$ generates samples via the inverse transformation of samples from the model prior. Then the discriminator $\mathcal{D}_\phi : \mathbb{R}^D \to \mathbb{R}$ is trained to distinguish between generated samples and real samples from the target distribution, with trainable parameters denoted ϕ. Various objectives can be used in training a GAN; here the Wasserstein GAN objective is used [15], which is expressed as

$$\min_{\theta}\max_{\phi} \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}}\left[\mathcal{D}_\phi(\mathbf{x})\right] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{Z}}}\left[\mathcal{D}_\phi(\mathcal{G}_\theta(\mathbf{z}))\right]. \tag{13}$$

Extending these objectives to the case of a conditional flow-based model, let the training data consist of tuples of features and labels, $(\mathbf{x}_n, \mathbf{c}_n)$, $n = 1,\ldots,N$. The maximum-likelihood loss can then be expressed as

$$J_{ml} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{D}\left[\log(p_{\mathbf{Z}_i}(f(\mathbf{x}_n,\mathbf{c}_n)_i)) + \log(|S_{ii}|)\right]. \tag{14}$$

To calculate the GAN loss, the discriminator takes as input both the features and labels, $[\mathbf{x}_n \quad \mathbf{c}_n]$. The discriminator loss is then

$$J_{disc} = -\frac{1}{N}\sum_{n=1}^{N}\mathcal{D}_\phi([\mathbf{x}_n \quad \mathbf{c}_n]), \tag{15}$$

and the generator loss is

$$J_{gen} = -\frac{1}{N}\sum_{n=1}^{N}\mathcal{D}_\phi([\mathcal{G}_\theta(\mathbf{z}_n,\tilde{\mathbf{c}}_n) \quad \tilde{\mathbf{c}}_n]), \tag{16}$$

where $\mathcal{G}_\theta(\mathbf{z}_n,\tilde{\mathbf{c}}_n) = f^{-1}(\mathbf{z}_n,\tilde{\mathbf{c}}_n)$, $\mathbf{z}_n \sim \mathbf{Z}$, and $\tilde{\mathbf{c}}_n \in \mathbf{C}$. In practise, the values of $\tilde{\mathbf{c}}$ are randomly sampled from $\mathbf{C}$.

Suppose the dataset consists of both labelled samples, $(\mathbf{x}_n^l, \mathbf{c}_n)$, $n = 1,\ldots,N_l$ and unlabelled samples $(\mathbf{x}_n^u)$, $n = 1,\ldots,N_u$ where N_l and N_u denote the number of labelled and unlabelled samples respectively. The maximum likelihood loss for labelled samples, J_{ml}^l remains the same as in (14). For unlabelled samples, the variable $\mathbf{c}$ for each sample is assigned the value that gives the maximum likelihood:

$$\mathbf{c}_n^u = \arg\max_{\mathbf{c}\in\mathbf{C}}\left[\log(p_{\mathbf{Z}_i}(f(\mathbf{x}_n^u,\mathbf{c})_i)) + \log(|S_{ii}|)\right]. \tag{17}$$

When $\mathbf{C}$ is a set of categorical variables, the maximum argument can be found by enumerating over all values of

C. Then the loss for unlabelled samples is

$$J_{ml}^{u} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{D}\left[\log(p_{\mathbf{Z}_i}(f(\mathbf{x}_n^u, \mathbf{c}_n^u)_i)) + \log(|S_{ii}|)\right]. \tag{18}$$

The overall loss function for training by example combines the maximum likelihood and GAN objectives by summing their losses:

$$J_{total} = J_{ml}^{l} + J_{ml}^{u} + J_{gen}. \tag{19}$$

2.4 Training Procedure

Algorithm 1 shows the procedure for training the conditional flow-based model with labelled and unlabelled training samples. At each training epoch, the discriminator is trained for N_{disc} iterations. The number of discriminator iterations per epoch is set as $N_{disc} = 5$, as used in other works [14].

Algorithm 1 Flow-based model semi-supervised training

1: Input: training data of labelled true points $(\mathbf{x}_n^l, \mathbf{c}_n)$ and unlabelled true points $(\mathbf{x}_n^u)$
2: initialise model (generator) parameters θ, discriminator parameters ϕ, and training data $\mathbf{X}_{train}$
3: **for** epoch $= 1$ to N_{epoch} **do**
4: sample generated points $(\mathcal{G}_\theta(\mathbf{z}, \tilde{\mathbf{c}}_n), \tilde{\mathbf{c}}_n)$ from the model
5: **for** disc. epoch $= 1$ to N_{disc} **do**
6: train discriminator parameters ϕ on true points and generated points
7: **end for**
8: train model parameters θ on batches of training data
9: **end for**

3 Experiments

3.1 Five Body Problem

We use the formulation of the CR5BP originally introduced by Ollöngren [16] and later studied in terms of its basins of attraction [6]. The problem consists of four primary bodies $P_i,\ i = 0, 1, 2, 3$ with dimensionless masses $m_0 = \beta m$, $m_1 = m_2 = m_3 = m = 1$. To model the fifth body, the system uses a rotating frame of reference with origin at the centre of mass of the primary bodies. The positions of the centres of the primary bodies are $(x_0, y_0) = (0, 0)$, $(x_1, y_1) = (1/\sqrt{3}, 0)$, $(x_2, y_2) = (-1/(2\sqrt{3}), 1/2)$, $(x_3, y_3) = (-1/(2\sqrt{3}), -1/2)$. The fifth body has negligible mass and its position is denoted (x, y).

As defined by Ollöngren [16], the time-independent effective potential function of this problem is

$$\Omega(x, y) = \frac{1}{2}\left(x^2 + y^2\right) + k\sum_{i=0}^{3}\left[\frac{m_i}{r_i}\right], \tag{20}$$

where

$$k = \frac{1}{3(1+\beta\sqrt{3})}, \tag{21}$$

$$r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2},\ i = 0, 1, 2, 3. \tag{22}$$

The partial derivatives of this potential function are

$$\frac{\partial\Omega}{\partial x} = x - k\sum_{i=0}^{3}\frac{m_i(x - x_i)}{r_i^3}, \tag{23}$$

$$\frac{\partial\Omega}{\partial y} = y - k\sum_{i=0}^{3}\frac{m_i(y - y_i)}{r_i^3}. \tag{24}$$

The critical points that represent equilibrium solutions of the system are defined by the points where $\frac{\partial\Omega}{\partial x} = 0$ and $\frac{\partial\Omega}{\partial y} = 0$.

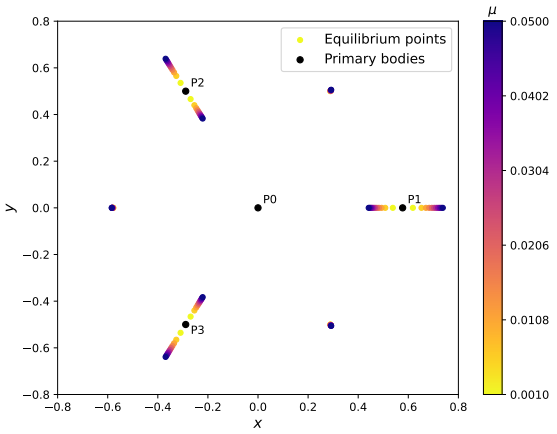


Figure 1: Locations of equilibrium points for a range of values of mass parameter μ.

Defining a mass parameter $\mu = 1/(1 + \beta)$, the value of μ controls the number of equilibrium points, their relative position, and their stability characteristics. In the case where $\mu >= 0.98617275$ there are 15 equilibrium points and for values of μ lower than this there are 9. This work considers values of the mass parameter with 9 equilibrium points since these are more interesting in terms of

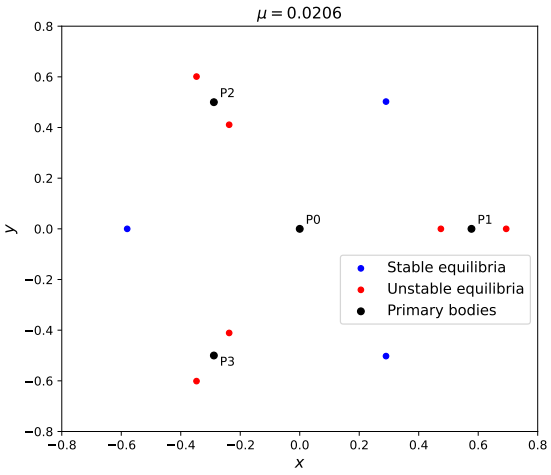


Figure 2: Locations of stable and unstable equilibrium points where the mass parameter $\mu < \mu_{crit}$.

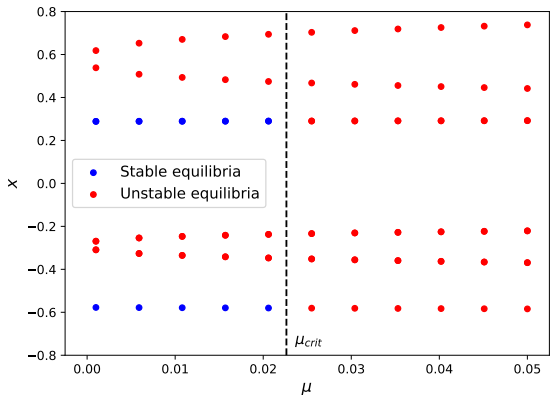
the stability of the points. Fig. 1 shows the location of equilibrium points and primary bodies over a range of μ values between 0.001 and 0.05.

The mass parameter of $\mu_{crit} = 0.02263467$ is a critical value that affects the stability of the equilibrium points. Above this value, all equilibrium points are unstable. Below this value, 3 of the 9 equilibrium points are stable as shown in Fig. 2.

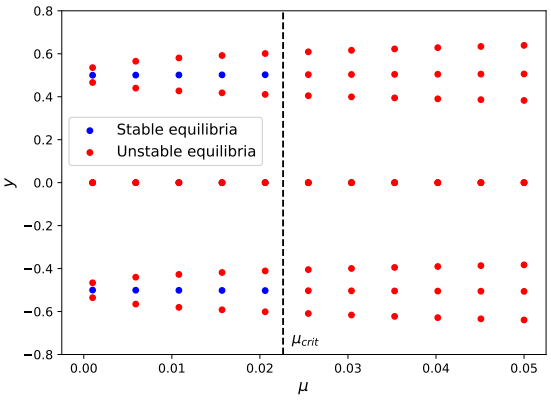
3.2 Training Data

The training data for this problem are known locations of equilibrium points for certain values of the mass parameter μ. The number of discrete mass parameters used in training is denoted N_μ. For this scale of problem, these points can be found using conventional root-finding methods as shown, for example, in [6]. Then the aim of training a conditional flow-based model on these data is to generalise to any value of μ within a specified range and generate points with certain stability characteristics. The training dataset consists of 8192 points and their relevant labels that are oversampled from the known equilibrium points.

Fig. 3 shows the training data generated for 11 evenly spaced values of μ in the range $\mu \in [0.001, 0.05]$. The training data contain the location of all minima for each value of μ as well as their stability class label. In this case, with 11 discrete values of μ, the problem effectively becomes one of supervised learning where the model can straightforwardly replicate the pattern in the training data.



(a) μ vs x

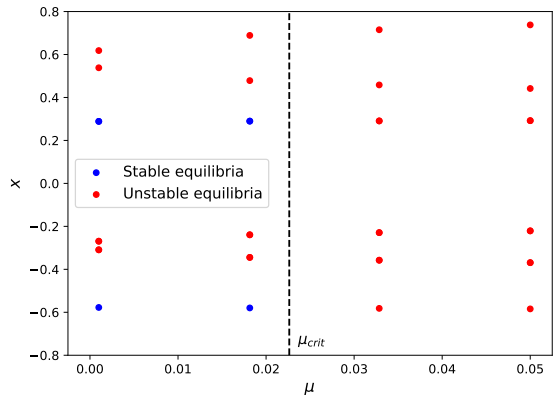


(b) μ vs y

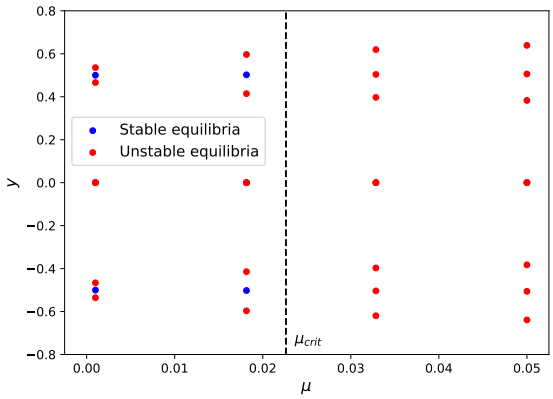
Figure 3: Training data with 11 discrete values of mass parameter μ and labels according to stability.

Fig. 4 shows the training data generated for only 4 values of μ, which represents a more difficult learning task. As before, the dataset has all minima for each value of μ and the stability class label. In this case, the gaps between values of the mass parameter make it more difficult for the model to generalise to unseen values, as will be shown.

The experiments shown here use two different formulations of physical variables, $\mathbf{x}$ and conditional variables $\mathbf{c}$. In the first case, the mass parameter μ is the conditional variable. This is normalised to be in the range $[-1, 1]$ as indicated in (25). In this case, the model does not account for the stability characteristics of these points. The other formulation has the mass parameter included as a physical variable with the conditional variables indicating the stability of the equilibrium point as shown in (26). The stability class is encoded as a one-hot vector where $[0\ 1]$ corresponds to a stable point and $[1\ 0]$ corresponds to an



(a) μ vs x



(b) μ vs y

Figure 4: Training data with 4 discrete values of mass parameter μ and labels according to stability.

unstable point.

$$\mathbf{x} = [x\ y]\,, \quad \mathbf{c} \in [-1, 1] \tag{25}$$

$$\mathbf{x} = [x\ y\ \mu]\,, \quad \mathbf{c} \in \{[0\ 1],\ [1\ 0]\} \tag{26}$$

4 Results and Discussion

This section shows the results of training flow-based generative models on the problems described above. When training on 11 discrete mass parameters, the model was trained for 1000 epochs. When training on only 4 discrete mass parameters, the model is trained for 100 or 500 epochs to reduce overfitting, as will be discussed. In each training case, the number of discriminator iterations per epoch is $N_{disc} = 5$. Table 1 shows the hyperparameters used in training. The values listed for the Generator indicate values for the m function used in each of the four coupling layers.

Table 1: Hyperparameters for training the discriminator and generator DNNs.

Hyperparameter	Value	
	Generator	Discriminator
N hidden layers	3	3
N hidden units	512	256
Learning rate	0.001	0.001
Batch size	256	256

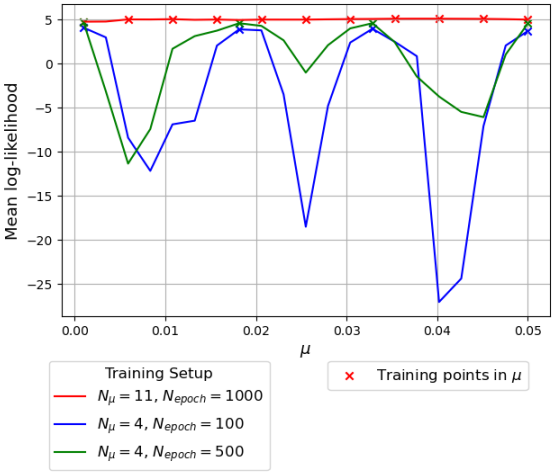


Figure 5: Mean log-likelihood of equilibrium points of model trained with mass parameter as conditional variable.

4.1 Mass parameter conditional variable

The initial model training uses the mass parameter μ as the conditional variable. Fig. 5 shows the mean log-likelihood of equilibrium points across different mass parameters for each of the trained models. Higher log-likelihoods are desirable since this indicates a higher probability of the model sampling at or near these points. When training with $N_\mu = 11$, the model learns a mapping with high log-likelihood across the range of mass parameters. However, when reduced to $N_\mu = 4$, the values of mass parameter outwith the training data show significantly lower log-likelihoods. Nevertheless, even when the log-likelihood of certain points are lower on average, the model can still effectively converge on all minima over the range of mass parameters using conventional root-finding approaches. This is shown in Fig. 6, which compares the mean number of function evaluations to find all minima from model samples to random uniform samples. Note that for the model samples, points are only shown for values of the mass parameter outwith those used in

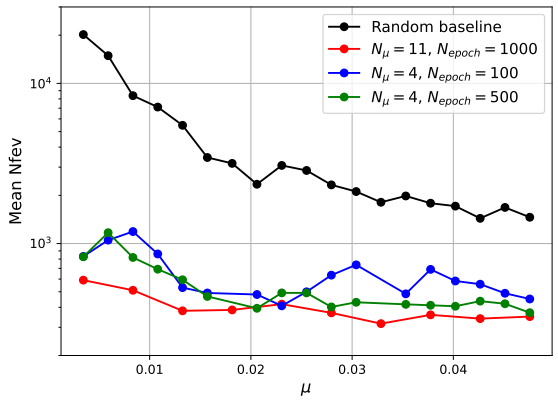
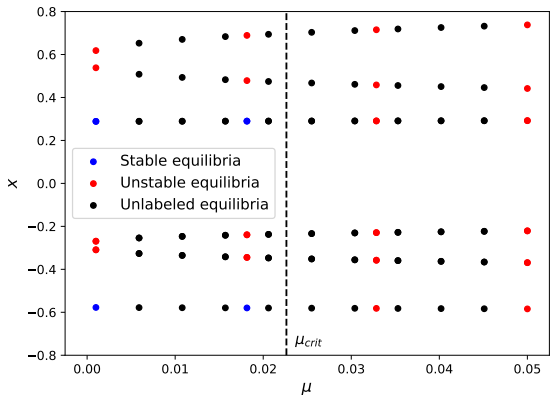
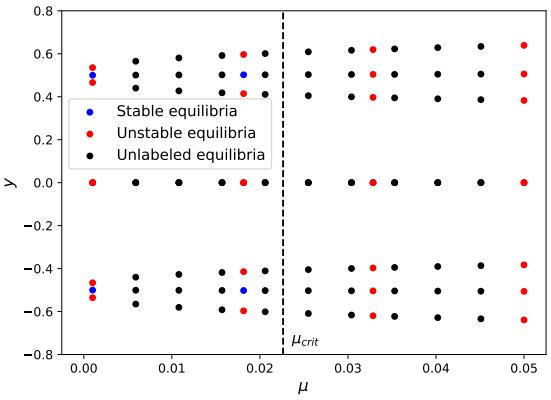


Figure 6: Mean number of function evaluations to find all equilibrium points for a given mass parameter.



(a) μ vs x



(b) μ vs y

Figure 7: Training data with 4 discrete values of mass parameter μ and labels according to stability.

training. As expected, sampling from the model consistently requires fewer function evaluations than sampling uniformly over the physical space.

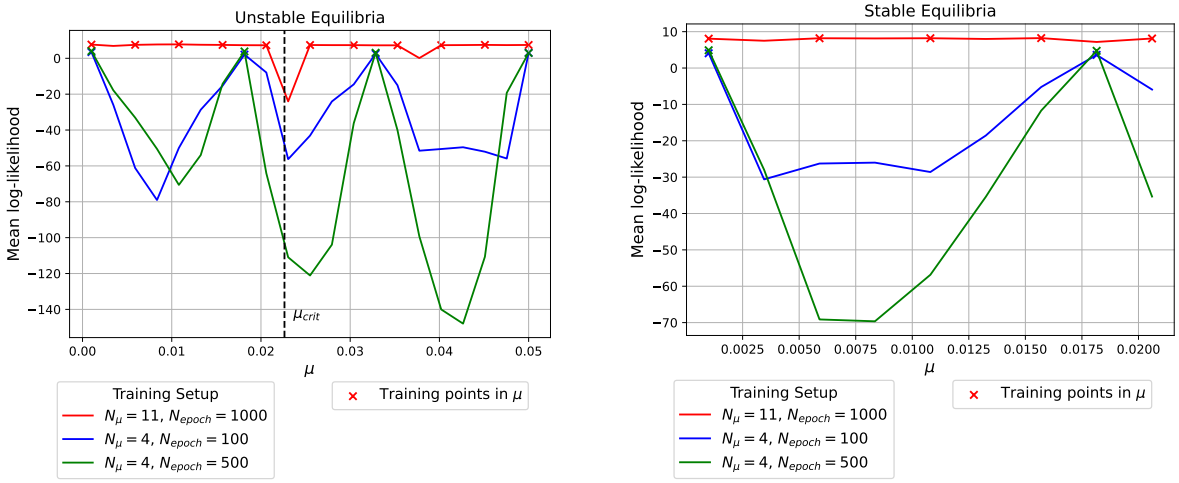
Samples generated and converged to equilibrium points by the model conditioned on mass parameter can be used to generate unlabelled training data. This is shown in Fig. 7 where unlabelled samples effectively fill the gaps in mass parameter between labelled samples where $N_\mu = 4$. Using these additional points allows semi-supervised training of a model conditioned on stability, which has some advantages as will be shown.

4.2 Stability conditional variable

All of the following models are trained using the stability as conditional variable and the position and mass parameter as physical variables as described by (26). Initially, the models are trained only using labelled training data. Fig. 5 shows the mean log-likelihood of equilibrium points across different mass parameters for both stable and unstable points. Similarly to the model conditioned on mass parameter, the model trained with $N_\mu = 11$ has a higher log-likelihood across the range of mass parameters than those trained with $N_\mu = 4$. This is the case for both the unstable and stable equilibria. However, the unstable equilibria see a decrease in log-likelihood for values of μ slightly larger than μ_{crit}. This is due to the model having less certainty on the correct class of points in this region. Compared to the mass parameter conditional model, these models show far more significant dips in log-likelihood for mass parameters not in the training data. This effect is more pronounced when training for a larger number of epochs, which indicates overfitting to certain mass parameters can be an issue in these models.

To visualise how well these models generalise over values of mass parameter, we can observe the distribution of samples generated by the model. Fig. 9 shows this as a Gaussian kernel density estimate over values of mass parameter for 1000 samples of each class from the trained models. Clearly, the model with $N_\mu = 11$ has the most even distribution of points in mass parameter. However, it also creates more samples with a stable class label above the critical mass parameter, which cannot be stable. As expected, the models with $N_\mu = 4$ have a much higher density of samples near points within the training data, which again highlights their tendency to overfit to previously seen points.

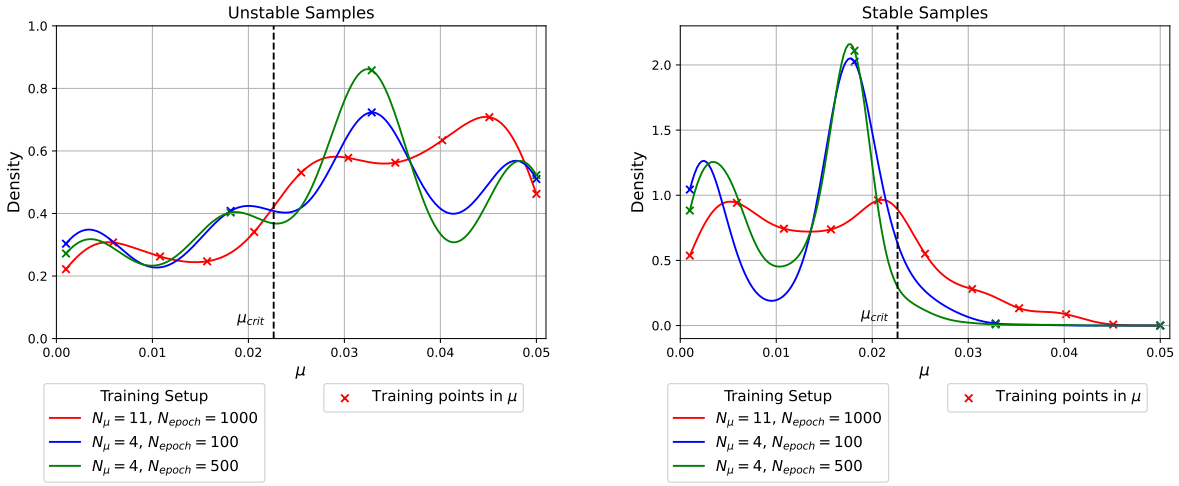
The trained models can also be used for classification by observing which class gives the highest log-likelihood for a given point in physical space. Each model is tested by classifying points over a range of 21 discrete mass pa-



(a) Unstable class

(b) Stable class

Figure 8: Mean log-likelihood of equilibrium points of model trained with stability class as conditional variable.



(a) Unstable class

(b) Stable class

Figure 9: Kernel density estimate over mass parameter of 1000 sampled points from models trained with stability class as conditional variable.

rameters. This gives a testing set of 189 points of which 27 are stable and 162 unstable. The classification performance of each of the trained models is shown in Table 2, Table 3, and Table 4.

Table 2: Confusion matrix of classes for conditional model trained with $N_\mu = 11$.

		Predicted Class		
		Stable	Unstable	Total
True Class	Stable	27	0	27
	Unstable	2	160	162
	Total	29	160	

Table 3: Confusion matrix of classes for conditional model trained with $N_\mu = 4$, $N_{epoch} = 100$.

		Predicted Class		
		Stable	Unstable	Total
True Class	Stable	26	1	27
	Unstable	37	125	162
	Total	63	126	

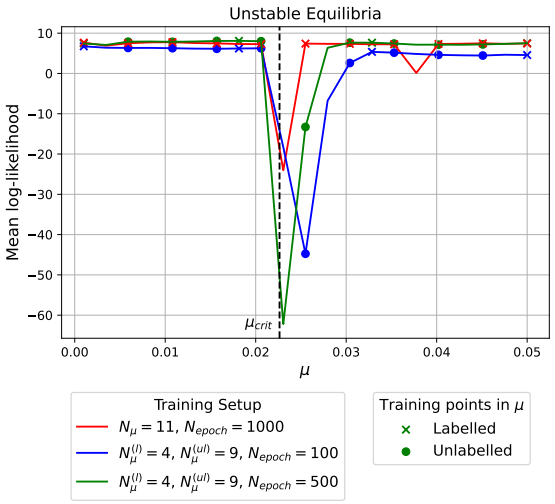
Table 4: Confusion matrix of classes for conditional model trained with $N_\mu = 4$, $N_{epoch} = 500$.

		Predicted Class		
		Stable	Unstable	Total
True Class	Stable	27	0	27
	Unstable	34	128	162
	Total	61	128	

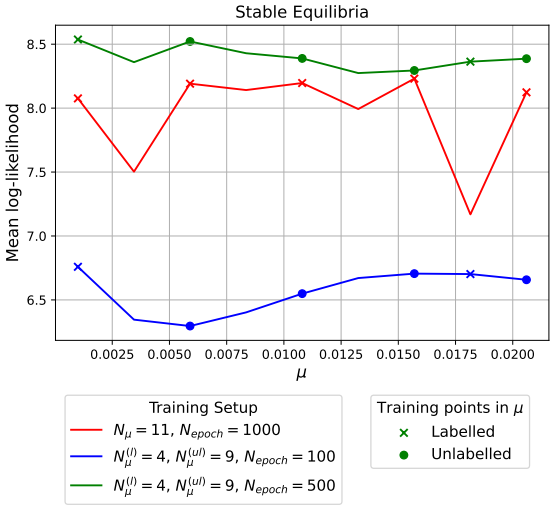
As in previous cases, the model trained with $N_\mu = 11$ has the best classification performance with only two points classed incorrectly as stable. Both models trained with $N_\mu = 4$ tend to over-classify points as stable and have a similar overall classification performance.

4.3 Semi-supervised training

The case where equilibrium points are given for only 4 discrete mass parameters represents a more realistic scenario for more complex dynamical systems for which these points are difficult to calculate. As shown in the results above, the performance of models trained with only these points is poor. One possible way to improve the models is to train semi-supervised using the data shown in Fig. 7. The following results are from models trained in a semi-supervised manner over 100 and 500 epochs with the stability class as conditional variable. Results are compared to the model trained with $N_\mu = 11$.



(a) Unstable class

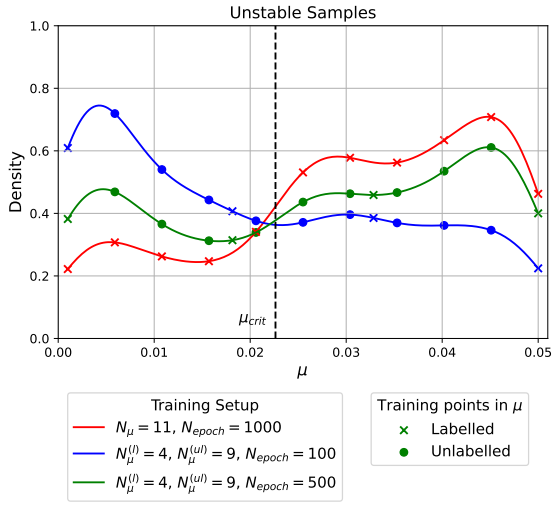


(b) Stable class

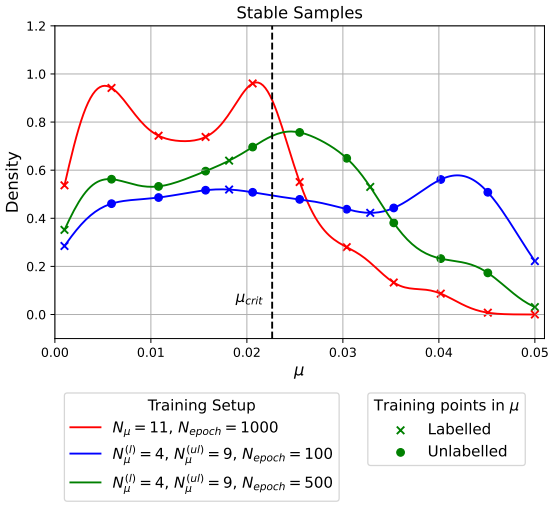
Figure 10: Mean log-likelihood of equilibrium points of model trained semi-supervised with stability class as conditional variable.

The mean log-likelihood over mass parameter for these models is shown in Fig. 10. As with the other models, there is a noticeable dip in likelihood for unstable equilibria with mass parameters slightly higher than the critical value. However, the log-likelihood does not decrease as substantially as in the case with only $N_\mu = 4$. Over most values of mass parameter, the mean log-likelihood is similar to that of the model trained with $N_\mu = 11$. In fact, in some cases the model trained semi-supervised for 500 epochs has better performance in terms of log-likelihood than the baseline model. Both of the semi-supervised models clearly perform better in this regard than the mod-

els trained with $N_\mu = 4$.



(a) Unstable class



(b) Stable class

Figure 11: Kernel density estimate over mass parameter of 1000 sampled points from models trained semi-supervised with stability class as conditional variable.

Now considering the samples across mass parameter, Fig. 11 shows the kernel density estimate over mass parameter of sampled points from the semi-supervised models. Compared to the models trained without unlabelled points, the samples are distributed far more evenly in mass parameter. However, they also have a tendency to generate more unphysical samples labelled as stable with mass parameters higher than μ_{crit} - particularly for the model trained for 100 epochs. This suggests that the models trained with unlabelled samples have greater difficulty learning the distribution of classes as would be expected.

Table 5: Confusion matrix of classes for semi-supervised conditional model with $N_{epoch} = 100$.

		Predicted Class		
		Stable	Unstable	Total
True Class	Stable	27	0	27
	Unstable	32	130	162
	Total	59	130	

Table 6: Confusion matrix of classes for semi-supervised conditional model with $N_{epoch} = 500$.

		Predicted Class		
		Stable	Unstable	Total
True Class	Stable	27	0	27
	Unstable	15	147	162
	Total	42	147	

Table 5 and Table 6 show the confusion matrices for the semi-supervised models. Despite their sampling performance, these models perform better in classification than the other models trained with $N_\mu = 4$. The model trained semi-supervised shows the best performance with 15 points misclassified as stable - although this is still worse than the model trained with $N_\mu = 11$. This shows that unlabelled examples can also improve the classification ability of a model.

5 Conclusions

This work introduced a new method for the joint generation and classification of critical points using conditional flow-based models. Conditional variables can be a physical parameter of the system or a characteristic of the critical points. The proposed approach allows the model to learn in a semi-supervised manner to generate points with a specified stability characteristic. Compared to training a model on only labelled points, the semi-supervised training using samples from another model improved the likelihood of sampling equilibrium points across the range of mass parameters.

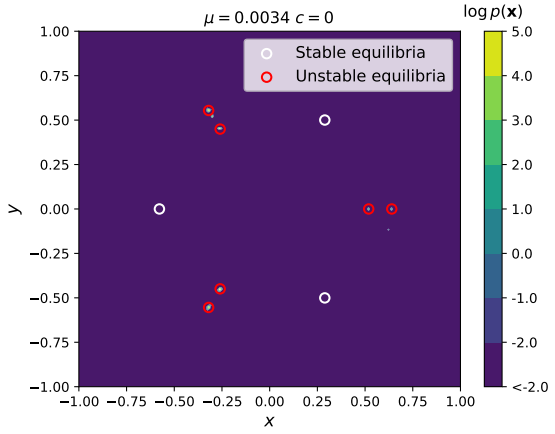
The application considered here of equilibrium points in the CR5BP is small scale to serve as a demonstration. Future work will investigate how well this approach can scale to more complex systems. In addition, further work is necessary to unify the two approaches of conditioning the flow-based model on both physical parameters and stability characteristics.

Acknowledgements

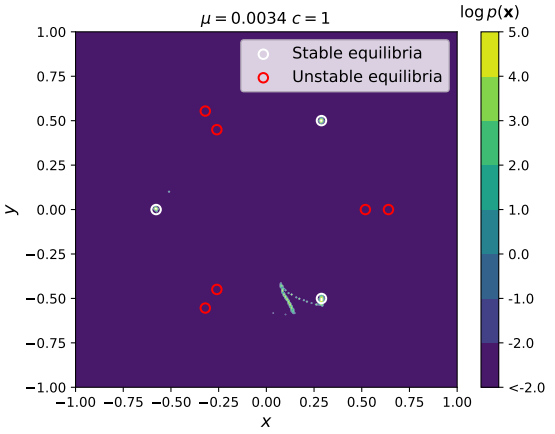
The results in this paper were generated within project GENEPY (Generative Mapping and Control of Stationary Points in Complex Dynamical Systems), EPSRC reference EP/X018288/1.

Appendix A - Model log-likelihood in physical space

The following figures give visualisations of the density predicted by some of the trained models. These are shown for a value of the mass parameter not seen by the model during training. Figs. 12, 13, and 14 show these plots for the models trained with stability conditional variable, semi-supervised stability conditional variable, and mass parameter, respectively.



(a) Unstable class label



(b) Stable class label

Figure 12: Log-likelihood in physical space for the model trained with stability class as conditional variable, $N_\mu = 11$, $N_{epoch} = 1000$.

References

[1] L. Ruthotto and E. Haber, "An introduction to deep generative modeling," *GAMM-Mitteilungen*, vol. 44, no. 2, 2021.

[2] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 1316–1324.

[3] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-Shot Text-to-Image Generation," Feb. 2021, arXiv:2102.12092 [cs]. [Online]. Available: http://arxiv.org/abs/2102.12092

[4] F. Verhulst, *Nonlinear Differential Equations and Dynamical Systems*. Springer Science & Business Media, Feb. 2006.

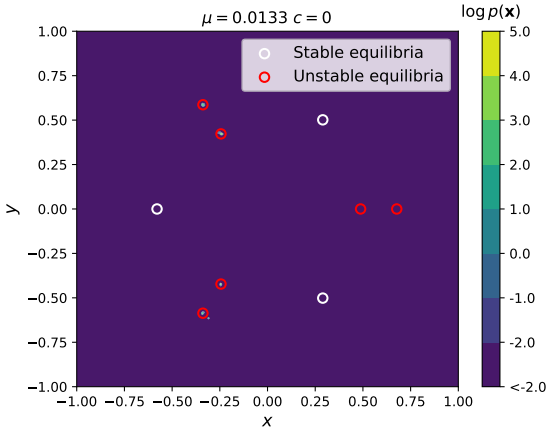
[5] D. J. Scheeres, "Stability of the planar full 2-body problem," *Celestial Mechanics and Dynamical Astronomy*, vol. 104, no. 1, pp. 103–128, Jun. 2009.

[6] E. E. Zotos and S. Suraj, "Basins of attraction of equilibrium points in the planar circular restricted five-body problem," *Astrophysics and Space Science*, vol. 363, pp. 1–16, 2018.

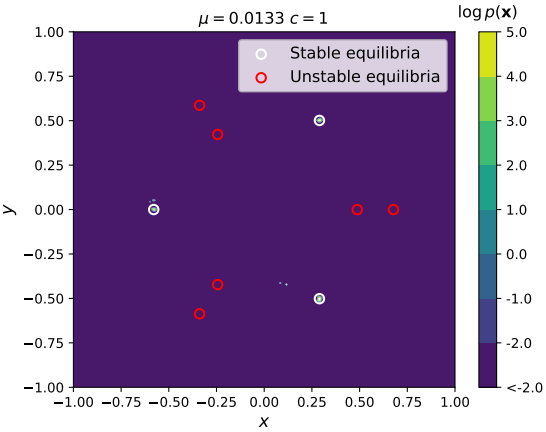
[7] A. Ghadami and B. I. Epureanu, "Data-driven prediction in dynamical systems: recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 380, no. 2229, Jun. 2022.

[8] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images," in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015.

[9] K. Parsopoulos and M. Vrahatis, "Computing periodic orbits of nondifferentiable/discontinuous mappings through particle swarm optimization," in *Proceedings of the 2003 IEEE Swarm Intelligence Sym-*



(a) Unstable class label



(b) Stable class label

Figure 13: Log-likelihood in physical space for the model trained semi-supervised with stability class as conditional variable, $N_{epoch} = 500$.

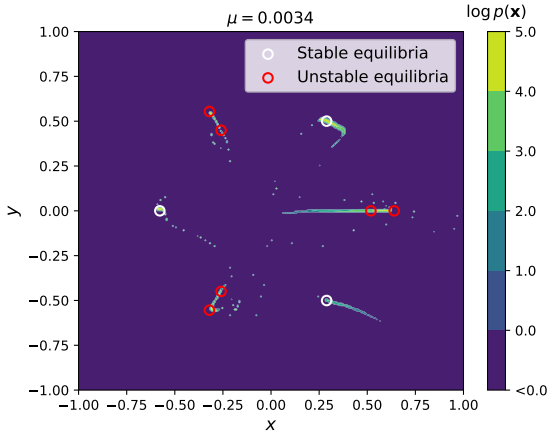


Figure 14: Log-likelihood in physical space for the model trained with mass parameter as conditional variable, $N_{\mu} = 4$, $N_{epoch} = 500$.

posium. SIS'03 (Cat. No.03EX706). Indianapolis, IN, USA: IEEE, 2003, pp. 34–41.

[10] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing Flows: An Introduction and Review of Current Methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3964–3979, Nov. 2021.

[11] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 2617–2680, Jan. 2021.

[12] L. Dinh, D. Krueger, and Y. Bengio, "NICE: Nonlinear Independent Components Estimation," Apr. 2015, arXiv:1410.8516 [cs]. [Online]. Available: http://arxiv.org/abs/1410.8516

[13] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data," *Journal of Computational Physics*, vol. 394, pp. 56–81, Oct. 2019.

[14] A. Grover, M. Dhar, and S. Ermon, "Flow-GAN: Combining Maximum Likelihood and Adversarial Learning in Generative Models," Jan. 2018, arXiv:1705.08868 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1705.08868

[15] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein Generative Adversarial Networks," in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, Jul. 2017, pp. 214–223, iSSN: 2640-3498.

[16] A. Ollongren, "On a particular restricted five-body problem an analysis with computer algebra," *Journal of Symbolic Computation*, vol. 6, no. 1, pp. 117–126, Aug. 1988.