University of Padova

Department of Physics and Astronomy "Galileo Galilei"

Master Thesis in Physics of Data

A new machine-learning framework to generate star cluster models

Internal Supervisor

Dr. Giuliano Iorio
University of Padova

External Supervisors

Dr. Mario Pasquato
CIELA institute, Montréal, Canada

Prof. Michela Mapelli
Heidelberg University, Germany

Master Candidate

George-Pantelimon Prodan

Student ID

2046802

Academic Year

2022-2023

Abstract

The birthplaces of stars are complex places, where turbulent interstellar gas collapses and fragments into star-forming cores, giving rise to non-trivial substructure. While the formation process can be modelled with hydrodynamical simulations, these are quite expensive in terms of computational resources. Moreover, primordial star clusters that are still embedded in their parent gas cloud are hard to constrain observationally. In this context, most efforts aimed at simulating the dynamical evolution of star clusters assume simplified initial conditions, such as truncated Maxwellian models.

We aim to improve on this state-of-the-art by introducing a set of tools to generate realistic initial conditions for star clusters by training an appropriate class of machine learning models on a limited set of hydrodynamical simulations. In particular, we will exploit a new approach based on Gaussian process (GP) models, which have the advantage of differentiability and of being more tractable, allowing for seamless inclusion in a downstream machine learning pipeline e.g. for inference purposes. The proposed learning framework is a two-step process including the model training and the sampling of new stellar clusters based on the inference results. We investigate different sampling approaches in order to find samplers that are able to generate realistic realizations.

Contents

Listing of figures

Listing of tables

Listing of acronyms

ML . . . . . Machine Learning

GP . . . . . Gaussian Process

MC . . . . . Monte-Carlo

MCMC . . . . . Markov Chain Monte-Carlo

NMMC . . . . . Non-Markovian Monte-Carlo

RWMC . . . . . Random Walk Monte-Carlo

EMCMC . . . . . Energy-based Markov Chain Monte-Carlo

APES . . . . . Approximate Posterior Ensemble Sampler

***k*-NN** . . . . . K-Nearest Neighbors

RBF . . . . . Radial Basis Function

CM . . . . . Center of Mass

NNLS . . . . . Non-negative Least Squares

GPU . . . . . Graphics Processing Unit

CPU . . . . . Central Processing Unit

GAN . . . . . Generative Adversarial Network

VAE . . . . . Variational Autoencoders

1 Introduction

The star clusters, groups of stars that are gravitationally bound together, are intensively studied by astronomers as they provide clues about the formation and evolution of galaxies. A cluster is a group of at least 12 stars whose volume is not dominated by dark matter [2]. The groups with fewer stars are called multiple star systems. There are two categories of bound clusters: open and globular. In the Milky Way, open clusters are located inside the galactic disc. They are young, low-mass clusters. On the other hand, globular clusters are found in the bulge and the halo of the galaxy. They are older and more massive than the open clusters [3].

Open clusters are born from the molecular clouds, the densest regions of the interstellar medium. These clouds are predominantly composed of molecular hydrogen. Their dynamics and hierarchical morphology are quite complex. There are two scenarios that aim to explain the mechanism of star formation: the gravoturbulent scenario and the Global Hierarchical Collapse scenario [3].

The gravoturbulent scenario states that the clouds manifest turbulence, which facilitates the gravitational equilibrium thanks to the supersonic and isotropic turbulence pressure. This leads to density fluctuations generating local collapses due to Jeans instability [3]. It is from these collapses that stars are born according to this scenario.

The second scenario focuses on collapses at multiple scales, starting with the larger cloud itself and moving to progressively smaller scales. In the gravoturbulent model, turbulence plays a central role in dictating where and when stars form, while in this case, the hierarchical nature of collapse across different scales plays a crucial role [3].

An aspect that is essential in the evolution of young open clusters is the initial phase space distribution of the stars. In the past years, hydro-dynamical simulations were proposed for creating new sets of initial conditions [1, 4]. However, the large amount of physical information needed in running a simulation makes the entire process expensive in terms of the computational time. Recently, simple machine learning (ML) techniques have been applied to the study of star clusters [5], [6]. By using ML algorithms, astronomers can emulate simulations of star clusters with less computational costs and still taking into account a variety of factors, such as the initial phase-space distribution, or their mass. Yet, there is still a lack of ML implementations in this field.

1.1 Modeling star clusters with Gaussian Processes

The focus of this thesis is on the Gaussian Processes (GPs) modeling of star clusters. GPs are a powerful tool for modeling complex systems, including star clusters. The goal of this work is to generate synthetic star clusters that closely resemble real ones.

Firstly, GPs are statistical models that estimate complex, nonlinear relationships between variables. In our case, these variables include the positions, velocities, and masses of individual stars. To generate new synthetic data we need first an adequate dataset. An option can be using data from hydro-dynamical simulations such those proposed in [1].

Next, it is essential that we establish a pertinent target for training our Gaussian Process model. We opt to train the model using the probability density function of the star position in its feature space. This strategy allows the predictions of novel targets to facilitate the generation of new star cluster configurations via the implementation of an efficient sampling methodology. The feature space depends on the use-case. A direct approach would be to learn from the seven-dimensional distributions of the star clusters, which include the mass, the positions and the velocity components.

Once we develop a model of the relationships between variables within the star cluster, we can use this model to generate synthetic star clusters. To do this, we experiment on using different sampling methods such as rejection sampling [7], Monte Carlo methods [8] and other complex algorithms. These methods are briefly described in the third section.

We emphasize the remarkable capability of Gaussian Processes in recreating the initial structure of the stellar clusters. The framework we propose comprises two parts: the GP modeling and the sampling process. Both parts are crucial in reaching our goal.

(a) Pleiades
Credits: NASA, ESA, AURA/Caltech, Palomar Observatory

(b) NGC 4755
Credits: NOAO/AURA/NSF, WIYN Consortium, Inc.

Figure 1.1: Open star cluster images

1.2 Hydrodynamical simulations of star clusters

Stars are created during the gravitational collapse of a massive molecular cloud. During this process, one can observe the evolution of an embedded stellar cluster [9] that is still surrounded by the primordial gas and dust, and the star formation is still possible. However, the cloud will be eventually dispersed and this process will lead to a young open cluster. These kind of clusters contain up to a few thousand stars and they are less gravitationally bound than the globular clusters. This is why open clusters are not necessarily spherical, showing various morphologies. For example, in Fig. 1.1 we see two open clusters having an irregular shape. In the first subfigure, there are the Pleiades, a young cluster hosting around 3000 stars. The second cluster is NGC 4755, one of the youngest clusters known.

The fractality observed in stellar cluster systems is a result of the formation process from the molecular cloud of origin. This property has been studied in the past to understand the mechanism of star formation [10]. However, when talking about fractality as a common feature of star clusters, it is worth mentioning that several studies have shown that clusters can exhibit different morphologies [11], [12]. As mentioned in [13], this could be a consequence of early dynamical interactions or the dependence of stellar cluster formation mechanisms on the local galactic environment.

Substructures of young clusters show complex kinematics [1], which can lead to a certain dynamical evolution facilitating the formation of interesting objects such as black holes, supernovae, or massive binary systems. Moreover, the formation of compact object mergers, namely

binary black holes or binary neutron stars, is a topic of current interest [14].

Fractality has been analyzed for the star clusters formed with hydrodynamical simulations in [15], [16] and many other studies. In these experiments, the clusters are created from a turbulent molecular cloud whose collapse can be simulated. These clusters form showing high fractality, but they lose substructures in time due to mergers or other relaxation processes [1].

A well-known parameter, Q, is used by many authors in this context as it is a way to measure fractality. By definition, Q is a quantitative measure that helps in distinguishing between a smooth overall radial density gradient and a multi-scale fractal sub-clustering [10]. To compute this parameter, one needs to obtain the Minimum Spanning Tree of the cluster. After that, Q is computed as the ratio between the mean edge length, $\overline{m}$, and the normalised correlation length, $\bar{s}$. In the same study [10], they established that substructured clusters have a Q-parameter smaller than 0.8.

Another star cluster property that has been studied is the rotation. It has been shown that embedded clusters exhibit rotation inherited from their parent cloud [1], [17], [18]. The rotation of the simulated substructures has been analysed in [1] and [11] by selecting regions of highest mass density and angular momentum and rescaling them to the center of mass.

Ballone et al [1] have ran 10 hydrodynamical simulations from turbulent molecular clouds. The turbulence in the simulation is modeled as a divergence-free Gaussian random velocity field, following the statistical properties described by the Burgers power spectrum. Additionally, each simulation run uses a different turbulence seed as its initial condition to explore different possible outcomes and understand the system's behavior under various starting conditions. In their study they state that all the simulated clusters are strongly substructured (high fractality) and that the rotation is observed for each simulation. They also have found that the rotation is more noisy for sub-clusters having fewer stars. In our study, we use the results of these simulations for the training and validation phases of our models trying to capture both of fractality and rotation properties.

1.3 Generating artificial clusters

Generating synthetic star cluster realizations can be a challenging task, especially when you take into account every parameter of each star (mass, coordinates, velocity components). A straightforward method of generating the positioning of the stars has been used by Cartwright and Whitworth in [10], where they use random numbers to generate artificial star clusters of 100-300 stars, but without establishing the velocity and the mass for each star. They create star

clusters of type "$3D\alpha$" using the following generative equations:

$$r = ((3-\alpha)\mathcal{R}_r/3)^{1/(3-\alpha)}, \tag{1.1}$$

$$\theta = cos^{-1}(2\mathcal{R}_\theta - 1), \tag{1.2}$$

$$\varphi = 2\pi\mathcal{R}_\varphi \tag{1.3}$$

By generating a set of random numbers between 0 and 1 ($\mathcal{R}_r, \mathcal{R}_\theta, \mathcal{R}_\varphi$), they obtain the polar coordinates of a star cluster targeting a volume density of $n \propto r^{-\alpha}$. This approach is reproducing clusters having spherical spatial distributions. Thus, it is not a suitable option if one aims for realistic realizations of young stellar clusters. As mentioned before, young, open stellar clusters can show different morphologies and sampling according to a power law volume density is constraining the cluster to a certain morphology.

Therefore, aiming to generate realistic synthetic clusters one needs to take into account reproducing different morphologies. Using artificial intelligence algorithms is a more suitable approach. In [5] it is proposed a hierarchical clustering algorithm used to generate new initial conditions for N-body runs without the necessity of running additional hydro-dynamical simulations. The algorithm identifies subclusters in the existing set of initial conditions, creating a tree-like representation of the stellar cluster structure. To generate new realizations, the model can modify nodes of the hierarchical tree. The key contribution of this approach is that it allows for the efficient generation of diverse sets of initial conditions using a simple but fast algorithm. To test the reliability of the generated sets of clusters, they have ran N-body simulations. They state that these simulations show a comparable evolution to the original clusters, making them valid initial conditions that can be used in further experiments.

However, clusters generated by [5] exhibit similarities at small scale, preserving the fractal dimension. Another limitation is that one can generate only a finite number of new realizations depending on the amount of training data. The novelty in our approach is that we can generate an unlimited number of new realizations as a benefit of learning a likelihood distribution during the GP model training. Also, this will lead to new realizations that enclose various small scale configurations.

2 Methods

2.1 Gaussian Processes

Gaussian Processes (GPs) are a type of probabilistic model that can be used to make predictions about the value of a function at a given point based on noisy observations of the function [19]. GPs can be used to model a wide range of phenomena, even when the available data is limited [20], making them a more suitable choice than the deep neural networks in this type of scenarios.

To generate new star cluster realization one can model the distribution of stars in a cluster as a function of the physical parameters that govern the cluster, such as the star masses, coordinates and velocities. GPs can learn the relationships between these parameters and the distribution of stars in the cluster in order to generate synthetic clusters that are consistent with the learned relationships.

There are several advantages of using GPs here. They can model complex, nonlinear relationships between variables and can account for uncertainty in the observations making them a powerful tool for modeling complex physical systems.

The GP model is defined by a mean function, $\mu(x) = \mathbb{E}[f(\mathbf{x})]$ and a kernel funcion $K(\mathbf{x}, \mathbf{x'}) = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x'}) - \mu(\mathbf{x'}))]$, where $f(\mathbf{x})$ is the target distribution and $\mathbf{x}$ are the input features [21],

$$f : X \to Y$$

The GP modeling is done through Bayesian inference. It allows us to update our beliefs about the parameters of the model in light of new observations. Given some observed data, Bayesian inference allows it to compute the posterior distribution over the model parameters. This posterior distribution reflects the updated beliefs about the parameters given the observed data and allows us to make predictions about the distribution of stars with different masses and locations in the phase-space. This helps in order to generate new synthetic realizations of the cluster that are consistent with the observed data.

The model parameters include the hyperparameters of the kernel function, which include the lengthscale, and amplitude of the covariance function. The noise level is also considered a trainable parameter. Further, I briefly describe these key parameters of the GP model:

- **The amplitude of the covariance function** is the scale of the variation in the target function that is captured by the model. The covariance function gives us the degree of similarity between stars (as input points) and, therefore, determines how much influence each star has in the predictions involving other stars from the same cluster. One of the most used kernel functions is the squared exponential (radial basis function - RBF kernel),

  $$K(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f \exp(-\frac{1}{2l^2}|\mathbf{x}_p - \mathbf{x}_q|^2)$$

  Let us consider $\mathbb{E}[f(\mathbf{x})] = 0$, then we can write the prior as

  $$p(f|\mathbf{x}, \theta) = \mathcal{N}(0, K(\mathbf{x}, \mathbf{x}))$$

- **The lengthscale**, l, determines the distance over which the covariance function between stars decays to zero. A shorter length scale captures fine-grained details in the data, while a longer scale length results in a smoother function that captures larger-scale trends [22].

- **The signal variance**, σ_f, can be considered the *vertical* scale length, while l measures the scale length on the horizontal direction. It measures the signal amplitude.

- **The noise**, σ_n, is typically modeled as a separate term in the target function. It is relevant as it takes into account errors. Usually, this noise term is a Gaussian distributed with zero mean and a fixed variance. The noise level behaves as a trainable parameter during the modeling process. We need to define a likelihood function that takes into account the noise,

  $$p(y|f) \sim \mathcal{N}(f, \sigma_n^2 I)$$

  The noise level is not established in advance, but estimated during training by minimizing the loss function. The loss function is obtained from the mapping function distribution, $p(y, f|\mathbf{x}, \theta) = p(Y|f)p(f|\mathbf{x}, \theta)$. One can marginalize this distribution to obtain

the expression of the loss function,

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log 2\pi$$

2.2 Density Estimator

In order to obtain the target distribution for training the GP model one can approximate it by computing the star density for each star location in the feature space. The k-nearest neighbors (k-NN) density estimator, proposed in [23], is a simple approach to estimate the density based on a sample of observations. The idea behind k-NN density estimation is to estimate the density of a point as a function of the distance between that point and its k-th nearest neighbor. A point is more likely to belong to a higher density region if it is closer to many other points than if it is far away from most points. An advantage of the k-NN density estimator is that it does not require any assumptions about the underlying distribution of the data.

The k-NN density estimator works as follows: given a sample of observations, the algorithm first calculates the distances between all pairs of points. Then, for each point in the sample, it finds the k-nearest neighbors based on the distance metric used (in our case, the Euclidean distance). Finally, the density of each point is estimated as:

$$d = \frac{k\Gamma(n/2+1)}{\pi^{n/2}r^n}$$

where n is the dimensionality, r is the distance from a given data point to its k-th nearest neighbor, and Γ is the gamma function. The dimensionality, n, signifies the total number of features or independent variables we use to describe each data point. When training a seven-dimensional GP model, the dimensionality corresponds to the number of parameters that characterize each star. Therefore, if we use seven distinct parameters for every star in our dataset, our analysis would occur in a seven-dimensional space, meaning that $n = 7$, and the density of each point will be estimated computing the volume of a seven-dimensional hyper-sphere of radius r.

2.3 Sampling Methods

The choice of sampling method is a key aspect when generating new realizations. We experiment with several Monte Carlo methods [8], [24],[7].

2.3.1 Markov Chain Monte-Carlo

Markov Chain Monte-Carlo (MCMC) sampling involves constructing a Markov chain whose stationary distribution is the target distribution. It generates samples by running the Markov chain for a sufficient number of steps and collecting samples from the chain. This method can be efficient if the Markov chain mixes well, but may be slow to converge to the stationary distribution. The key reason of choosing MCMC is the capability of generating correlated samples, which is essential in the context of the complex physical systems (i.e. star clusters). In such systems, parameters, like the positions and velocities of stars, are not independently dispersed but they are correlated based on underlying physical laws.

We implement MCMC sampling using the Metropolis-Hastings algorithm [25]. We provide the complete pseudo-code in Algorithm 2.1.

An important parameter of the Metropolis algorithm is the step size. It refers to the size of the random jumps taken in the feature space by the Markov Chain at each iteration. It determines how far the chain moves from the current state to the next state. The step size affects the acceptance rate of new samples and can have a significant impact on the convergence and efficiency of the algorithm. If the step size is too small, the chain will take a long time to explore the parameter space and the sampler may get stuck in local minima. On the other hand, if the step size is too large, the chain may not be able to effectively explore the space and the acceptance rate may be low. A good choice of step size balances these considerations to achieve a high acceptance rate and efficient exploration of the parameter space.

Notice that in this MCMC algorithm version we do not generate any sample if the proposed sample is rejected.

We explore several variants of this algorithm:

- **Random Walk MCMC (RWMC)** - this algorithm samples the step size from a normal distribution. The algorithm resambles a random walk.
- **Non-Markovian Monte-Carlo (NMMC)** - in this version we do not sample the new star using the features of the star generated previously, but we go back by a random number of iterations. This sampling approach is not Markovian anymore as we take

Algorithm 2.1 MCMC sampling with Metropolis-Hastings algorithm

Require: $s > 0, N \in \mathbb{N}$
  samples $\leftarrow$ empty array to store N samples
  counter $\leftarrow 1$
  samples$[0] \leftarrow x_0$
  prob $\leftarrow \mathcal{GP}($*samples*$[0])$ {Use GP model to predict the probability}
  oldProb $\leftarrow$ *prob*
  while *counter* $< N$
    newStar $\leftarrow \mathcal{N}($*samples*$[$*counter* $- 1], s)$
    prob $\leftarrow \mathcal{GP}($*newStar*$)$
    $r \leftarrow \mathcal{U}(0, 1)$
    if $r <$ *min*$(1,$ *prob*$/$*oldProb*$)$
      samples$[$*counter*$]$ $\leftarrow$ *newPoint*
      counter $\leftarrow$ *counter* $+ 1$
      oldProb $\leftarrow$ *prob*
    end if
  end while

into account the process history. So, in this new variant we choose each time a random star from the previously generated ones, and we use it to sample a new candidate star (see Alg. 2.2). Also, we implement the **random walk** version of this algorithm.

Algorithm 2.2 NMMC Algorithm

  r_idx $\leftarrow \mathcal{U}(0,$ *counter* $- 1)$ Generate a random integer
  new_star $\leftarrow \mathcal{N}($*samples*$[$*r_idx*$], s)$
  d $\leftarrow \mathcal{GP}($*new_star*$)$

2.3.2 Approximate Posterior Ensemble Sampler

Approximate Posterior Ensemble Sample (APES) is a novel sampling algorithm proposed by Viventi and Barroso in 2023 [24]. This approach exhibits faster convergence than traditional MCMC methods and it is scalable to higher dimensions making it an alternative solution for sampling complex target distributions.

Introduction

In traditional MCMC methods, at each step we sample a single n-dimensional candidate, $\mathbf{x}$. On the other hand, in APES we sample using ensembles of L candidates,

$$X = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_L})$$

Each point in the ensemble is considered a *walker* for which new proposals are made based on an approximate distribution of the target $\pi(x)$. The approximate distribution for the walker x_i', $\tilde{\pi}(x|X_{[i]})$, is conditioned on the set created using the other walkers, i.e. $X_{[i]}$ is obtained by removing the i-th walker from X.

Let us consider a walker, $\mathbf{x_i}$. If we update $\mathbf{x_i}$ to $\mathbf{x_i'}$, the new ensemble becomes

$$\mathbf{X}' = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_i'}, ..., \mathbf{x_L})$$

This means that one needs to compute the approximate distribution L times serially. However, APES aims to parallelize this sampling procedure.

Viventi and Barroso [24] propose that the ensemble could be divided in two parts of length $L/2$, such that $\mathbf{X} = (\mathbf{X_1}, \mathbf{X_2})$. In this way, we can compute the approximate distributions, $\tilde{\pi}(\mathbf{x}|\mathbf{X_1})$ and $\tilde{\pi}(\mathbf{x}|\mathbf{X_2})$, and use them to sample in parallel the walkers from both sub-ensembles.

The approximate distribution

A key aspect of this algorithm is computing the approximate distribution. Good ensembles will lead to an approximate distribution close to $\pi(x)$, meaning that the sampling procedure goes in the right way.

To compute $\tilde{\pi}(\mathbf{x}|\mathbf{X_e})$ with $e \in \{1, 2\}$, we firstly determine the Mahalanobis distances between the walker points as described in [24],

$$D_k^2(x, x_k) = (x - x_k)^T \mathbf{C_k^{-1}} (x - x_k) \tag{2.1}$$

for all $\mathbf{x_k} \in \mathbf{X_e}$, and, then,

$$\tilde{\pi}(\mathbf{x}|X_e) = \sum_k \frac{w_k}{h^n} K_k \Big[\frac{D_k(x, x_k)}{h} \Big] \tag{2.2}$$

where $\mathbf{C_k}$ is a positive definite square matrix that can be computed using an unbiased estimator

on the sub-ensemble, h is a *bandwidth* parameter, $\mathbf{w}$ a set of weights, and K is a kernel function.

The acceptance probability

At each iteration, the acceptance probability of the candidate, $\mathbf{y_i}$, is computed for every walker, $\mathbf{x_i}$ as

$$\alpha_i(\mathbf{X_e}, \mathbf{y_i}) = min\Big(1, \frac{\tilde{\pi}(\mathbf{x_i}|\mathbf{X_e})\pi(\mathbf{y_i})}{\tilde{\pi}(\mathbf{y_i}|\mathbf{X_e})\pi(\mathbf{x_i})}\Big) \tag{2.3}$$

where $\mathbf{X_e}$ is the sub-ensemble that does not contain the walker $\mathbf{x_i}$. If the sample is accepted, the walker is updated, otherwise it remains unchanged.

Algorithm description

A strength of APES is that we can implement this algorithm using parallelized loops. Each walker in a sub-ensemble can be updated independently from the other walkers of the same cluster, but at the same time it takes into account the approximated distribution $\tilde{\pi}$ by using the other sub-ensemble as a reference.

A pseudo-code of the sampling procedure is given in Algorithm 2.3. The algorithm takes as inputs the maximum number of iterations, N, the length of the ensemble, L, and the initial ensemble, X_0. Before each parallelized loop, we draw a new set of candidates,

$$\mathbf{Y_e^{t+1}} = \{\mathbf{y_i}| \text{ for each } i \text{ with } \mathbf{x_i} \in \mathbf{X_e}^t\}$$

The sub-ensemble is updated later after calculating the acceptance rates in the for loop.

2.3.3 Rejection Sampling

Rejection sampling involves generating samples uniformly at random from the sample space, and accepting them with a probability proportional to their density. This method can be inefficient if the density varies significantly across the sample space and it generates uncorrelated samples. Therefore, this method could be relevant as a baseline, as it should lead to lower performances than the other methods.

Algorithm 2.3 APES algorithm

Require: N, L, X_0

1: Separate the walkers in two sub-ensembles: X_1, X_2
2: $t \leftarrow 0$
3: **while** $t =< N$
4: Compute $\tilde{\pi}(\mathbf{x}|\mathbf{X}_2^{\mathbf{t}})$ and draw new candidates $\mathbf{Y}_1^{\mathbf{t+1}}$
5: **for** $\mathbf{x_i} \in \mathbf{X}_1$
6: Compute $\alpha_i(\mathbf{X}_2^{\mathbf{t}}, \mathbf{y_i^{t+1}})$
7: **end for**
8: Update $\mathbf{X}_1^{\mathbf{t+1}}$
9: Compute $\tilde{\pi}(\mathbf{x}|\mathbf{X}_1^{\mathbf{t+1}})$ and draw new candidates $\mathbf{Y}_2^{\mathbf{t+1}}$
10: **for** $\mathbf{x_i} \in \mathbf{X}_2$
11: Compute $\alpha_i(\mathbf{X}_1^{\mathbf{t+1}}, \mathbf{y_i^{t+1}})$
12: **end for**
13: Update $\mathbf{X}_2^{\mathbf{t+1}}$
14: $t \leftarrow t + 1$
15:
16: **end while**

3

Energy based Markov Chain Monte-Carlo

3.1 Introduction

Training a GP model in a seven-dimensional space might not be reliable for sampling star systems in close interaction, i.e. binary systems, as the probability to propose candidate stars very close to each other in the physical space is low. This would lead to erroneous modeling of the cluster at small scale.

A possible solution to this problem could be incorporating some physical laws or rules. We propose a novel approach based on sampling energy states of star pairs. An energy state takes into account the gravitational potential energy of the pair and the kinetic energies of both stars.

We aim our focus on the binary stars by selecting pairs of nearest neighbors. For each star, we select the nearest neighbor, and we continue creating another pair from its nearest neighbor. By selecting these pairs sequentially, we create a chain of pairs. Therefore, we obtain several pairs, $N-1$, where N is the number of stars in the cluster.

In this way, if one follows the chain of energy states that is created, there will be two types of pairs of stars: that are bound gravitationally, and that are not bound. Both are equally important in order to sample a realistic cluster.

If we consider all the pairs possible, we will find $N(N-1)/2$ pairs. Taking into account that $N \approx 3000$, then the GP model training will be costly computationally. Moreover, most of these pairs will be associated with states having low potential energies, close to zero and the

kinetic energies will become redundant. Thence, training on this enormous amount of data will not help in the sampling process.

3.2 Incorporated physics

Let us consider that our objective is to sample a stellar cluster with N stars. Each star σ_i will have a position vector $\vec{r}_i$, velocity $\vec{v}_i$, and mass M_i, with $i \in \overline{1, N}$. We aim to sample the stars one by one following a energy chain estimated by the GP model. We start with an initial star σ_1 for which we know $\vec{r}_1$, $\vec{v}_1$, and M_1. At the moment, we will consider that these features are chosen arbitrarily. We sample the next stars based on the estimated potential and kinetic energy. Thus, at step i, when sampling σ_j, we can write

$$U_{ij} = -G\frac{M_i M_j}{r_{ij}} \tag{3.1}$$

$$K_i = \frac{1}{2}M_i v_i^2 \tag{3.2}$$

Sampling from the probability distribution function given by $\mathcal{GP}(U_{ij}, K_i, K_j)$, one can find the modulus of the relative distance, $r_{ij} = |\vec{r}_j - \vec{r}_i|$ between two stars σ_i and σ_j. The sampling approach is Markovian, the current state is defined by the star σ_i. Thus, we can determine the following quantities:

$$M_j/r_{ij} = -\frac{U_{ij}}{GM_i} \tag{3.3}$$

$$r_{ij}v_j^2 = -\frac{2GM_i K_j}{U_{ij}} \tag{3.4}$$

As it is a stellar cluster, one can assume that there is a relationship between the velocity and the position vector. We can model this relationship using another GP model.

Thus, by using (Eq. 3.4) we can find the most probable position of σ_2 in the phase space sampling from $\mathcal{GP}(\vec{r}, \vec{v})$. Its mass is simply calculated using (Eq. 3.3) afterwards. However, this approach can be difficult to implement as we have to explore a six-dimensional space in order to solve Eq. 3.4.

Alternatively, we could sample from a mass distribution given by $\mathcal{GP}(M)$. In this way, we can calculate r_{ij} and v_j, and using $\mathcal{GP}(\vec{r}, \vec{v})$, we can find the most probable position of σ_2 by searching for the optimal directions of the vectors $\vec{r}$ and $\vec{v}$.

3.3 Algorithm description

There are several steps that we need to follow in order to implement Energy based Markov Chain Monte-Carlo (EMCMC).

3.3.1 Define an energy space

The energy space is defined by a set of energies – kinetic energies of each star (K_i and K_j) and the mutual potential energy of the two-star pair (U_{ij}). This forms a 3-dimensional energy space. The set of energies is constructed based on the nearest neighbors of the stars as showed in Algorithm 3.1.

Algorithm 3.1 Calculate energies

$N, D \leftarrow$ shape of *cluster*
$r \leftarrow$ array of coordinates
$v \leftarrow$ array of velocity magnitudes
$m \leftarrow$ array of masses
$distMatrix \leftarrow$ compute distance matrix from r, fill diagonal with ∞
$sortedCluster, U, K_1, K_2 \leftarrow$ initialize empty arrays
$currentStarIdx \leftarrow$ start from the star with minimal x+y+z
$visited \leftarrow$ initialize boolean array of length N as False
$i \leftarrow 0$
while $i < N - 1$
    $visited[currentStarIdx] \leftarrow True$
    $sortedCluster[i] \leftarrow cluster[currentStarIdx]$
    $nnDist \leftarrow distMatrix[currentStarIdx]$
    $nnDist[visited] \leftarrow \infty$
    $nnIdx \leftarrow argmin(nnDist)$ {index of the nearest neighbor}
    $dr \leftarrow nnDist[nnIdx]$ {distance to the nearest neighbor}
    $U[i] \leftarrow m[nnIdx] * m[currentStarIdx]/dr$
    $K_1[i] \leftarrow m[currentStarIdx] * v[currentStarIdx]^2/2$
    $K_2[i] \leftarrow m[nnIdx] * v[nnIdx]^2/2$
    $currentStarIdx \leftarrow nnIdx$
    $i \leftarrow i + 1$
end while

3.3.2 Gaussian Processes modeling

We define a dataset by gathering data for several stellar clusters and compute the energy state chains as described in the previous subsection. Then, we train several GP models for predicting the probability distributions of the star features in the energy space $\mathcal{GP}(U_{ij}, K_i, K_j)$ and the phase space $\mathcal{GP}(\vec{r}, \vec{v})$, and for the star masses $\mathcal{GP}(M)$.

3.3.3 Sampling procedure

An MCMC algorithm is implemented to draw samples from $\mathcal{GP}(U_{ij}, K_i, K_j)$. At each step of the MCMC, the current state of the system (i.e., the set of energies) is used to propose a new energy state. The new energy state is accepted or rejected based on a probability criterion that takes into account the energies of the current and proposed states and the physics of the star system.

In Algorithm 3.2 we present the pseudocode of the EMCMC sampling algorithm. The algorithm takes as inputs the step-size, s, the number of stars that we aim to sample, N, the GP model, $\mathcal{GP}(U_{ij}, K_i, K_j)$, and *findNewStar*, a function that incorporates the physics needed in finding a new star based on the sampled energy state, *energyState*, and the previous sampled star, *prevStar*. We consider the universal gravitational constant equal to 1, because its value does not affect the training and sampling process. The energy space is scaled before training, and rescaled before applying the physical equations incorporated in Algorithm 3.3.

The function *findNewStar* (see Algorithm 3.3) uses the mass and phase-space GP models, $\mathcal{GP}(M)$ and $\mathcal{GP}(\vec{r}, \vec{v})$. Also, we need a function, randomDirection (see Algorithm 3.4), that generates a random direction based on two random generated numbers, $\mathcal{R}_\theta$ and $\mathcal{R}_\varphi$, which are used to generate pairs of angles (θ, φ) that correspond to directions sampled uniformly on a three dimensional sphere. These directions are used to generate new candidates for the new star that we aim to sample.

We define two hyper-parameters, *maxJump* and N_c. The first parameter is used to control the maximum distance between two stars sampled consecutively. The other parameter, N_c, is the number of star candidates we generate randomly based on the values of $|\vec{r}_{ij}|$ and $|\vec{v}_j|$. We use $\mathcal{GP}(\vec{r}, \vec{v})$ afterwards to find the best candidate, the star having the most probable position in the phase-space.

Algorithm 3.2 EMCMC Algorithm

Require: $s > 0, N \in \mathbb{N}, \mathcal{GP}_{(U_{ij},K_i,K_j)}$, findNewStar

1: $samples \leftarrow$ empty array to store N samples
2: $energyStates \leftarrow$ empty array to store N energy states
3: $counter \leftarrow 1$
4: $samples[0] \leftarrow x_0$
5: $prob \leftarrow \mathcal{GP}_{(U_{ij},K_i,K_j)}(energyStates[0])$
6: $oldProb \leftarrow prob$
7: **while** $counter < N$
8: $newEnergyState \leftarrow \mathcal{N}(energyStates[counter-1], s)$
9: $newEnergyState \leftarrow energyStates[counter-1][2]$
10: $prob \leftarrow \mathcal{GP}_{(U_{ij},K_i,K_j)}(newEnergyState)$
11: $r \leftarrow \mathcal{U}(0,1)$
12: **if** $r < min(1, prob/oldProb)$
13: $energyStates[counter] \leftarrow newEnergyState$
14: $oldProb \leftarrow prob$
15: $newSample \leftarrow$findNewStar$(samples[counter-1], newEnergyState)$
16: **if** $newSample$
17: $samples[counter] \leftarrow newSample$
18: $counter \leftarrow counter + 1$
19: **end if**
20: **end if**
21: **end while**

Algorithm 3.3 findNewStar

Require: *prevStar*, *energyState*, $\mathcal{GP}(M)$, $\mathcal{GP}(\vec{r}, \vec{v})$, randomDirection, *maxJump*, N_c

$m_j \leftarrow MCMC(\mathcal{GP}_{(M)})$
$m_i, \vec{r}_i, \vec{v}_i \leftarrow prevStar$
$U_{ij}, K_i, K_j \leftarrow energyState$
$|\vec{r}_{ij}| \leftarrow \frac{m_i m_j}{U_{ij}}$
if $|\vec{r}_{ij}| > maxJump$
    return null
end if
$|\vec{v}_j| \leftarrow \sqrt{\frac{2K_j}{m_j}}$
candidates $\leftarrow$ empty array to store N_c candidates
for each *candidate* in *candidates*
    $\vec{\lambda}_{\vec{r}} \leftarrow$ randomDirection()
    $\vec{\lambda}_{\vec{v}} \leftarrow$ randomDirection()
    $\vec{r}_j = \vec{r}_i + |\vec{r}_{ij}|\vec{\lambda}_{\vec{r}}$
    $\vec{v}_j = |\vec{v}_j|\vec{\lambda}_{\vec{v}}$
    $candidate \leftarrow m_j, \vec{r}_j, \vec{v}_j$
end for
$probs \leftarrow \mathcal{GP}_{(\vec{r},\vec{v})}(candidates)$
$bestCandidateIdx \leftarrow argmax(probs)$
return $candidates[bestCandidateIdx]$

Algorithm 3.4 randomDirection

$\theta \leftarrow 2\pi\mathcal{R}_\theta$
$\varphi \leftarrow cos^{-1}(2\mathcal{R}_\varphi - 1)$
$x_{dir} \leftarrow sin(\varphi)cos(\theta)$
$y_{dir} \leftarrow sin(\varphi)sin(\theta)$
$z_{dir} \leftarrow cos(\varphi)$
return $[x_{dir}, y_{dir}, z_{dir}]$

4 Results

4.1 Processing pipeline

4.1.1 Dataset

We use a dataset of ten clusters generated by [1] using hydro-dynamical simulations of molecular clouds. Each cluster has a number of roughly 2500-4000 stars and an order of $10^4 M_\odot$ in the total mass. Table 4.1 summarizes the properties of these clusters. The dataset is split in a training set of seven clusters and a validation set of three clusters.

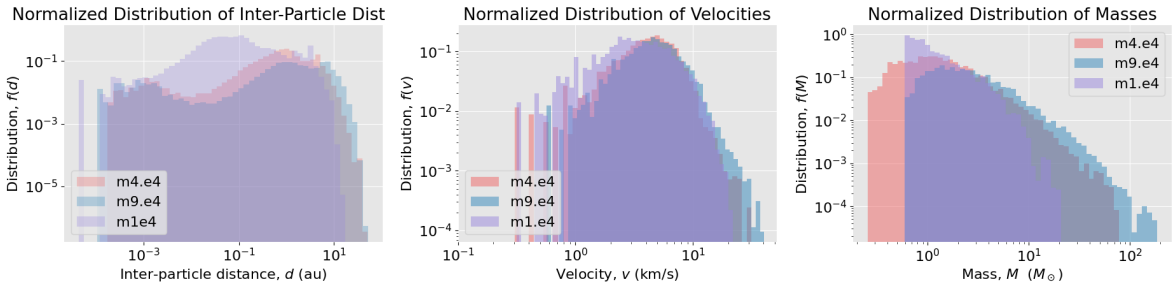
We show the inter-particle distance, velocity magnitude, and mass distribution for three clusters: m1e4, m4e4, and m9e4 in Fig. 4.1. We use these distributions as a baseline when evaluating the results obtained with our methods.

4.1.2 Models and training

To implement the models we use *gpytorch* library [26] which is an efficient and modular implementation of GPs supporting GPU acceleration. The GPU used in all the experiments is NVIDIA GeForce RTX 3060. We train the GP model using RBF kernel, exact marginal log likelihood for the loss function and Gaussian likelihood for computing the posterior distribution. A zero prior mean is used for all the models. Using simulations data we get the features needed for training our models. We train several GP models:

cluster	# stars	mass ($10^3 M_\odot$)	virial ratio	# binaries
m1e4	2523	4.2	1.19	400
m2e4	2571	6.7	1.32	515
m3e4	2825	10.3	1.48	545
m4e4	2868	14.4	1.47	546
m5e4	2231	14.1	1.47	490
m6e4	3054	20.4	1.69	671
m7e4	4214	31.5	1.50	797
m8e4	2945	28.3	1.60	561
m9e4	3161	30.5	1.52	666
m1e5	3944	38.0	1.46	757

Table 4.1: Statistics over the clusters obtained via hydrondynamical simulations by Ballone et al. [1]



(a) (b) (c)

Figure 4.1: Distributions of inter-particle distances (a), velocity (b), and mass (c) for three simulated clusters: m1e4, m4e4, and m9e4 [1]

model	best epoch	time/epoch (s)	l	σ_n
$\mathcal{GP}(M, \vec{r}, \vec{v})$	32	7.2	1.6205	0.1445
$\mathcal{GP}(\vec{r}, \vec{v})$	53	6.1	1.4506	0.0752
$\mathcal{GP}(U_{ij}, K_i, K_j)$	12	1.4	0.5034	0.3965
$\mathcal{GP}(M)$	69	1.0	1.5523	0.0040

Table 4.2: Training results. Columns: model (1), best epoch (2), training time per epoch (3), the lengthscale, l (4), the noise level, σ_n (5)

- $\mathcal{GP}(M, \vec{r}, \vec{v})$, a seven-dimensional model from which we can sample directly new clusters.
- $\mathcal{GP}(\vec{r}, \vec{v})$, a six-dimensional model that estimate the probability density function of the star position in the phase-space
- $\mathcal{GP}(U_{ij}, K_i, K_j)$, a three-dimensional model that estimate the probability density function of the energies stored by pairs of stars that are in close interaction
- $\mathcal{GP}(M)$, a one-dimensional model used to sample stellar masses

After defining the features for each model, we have to calculate the target distribution which is a probability density distribution. We use k-NN density estimators to determine the target distributions. Then, the model is trained and used in the sampling procedure afterwards. An overview of the processing pipeline that we develop is presented in Figure 4.2.

We train each model using cross-validation [27] and early stopping [28] with a patience of 10 epochs. In Figure 4.3 we show the training and validation losses with respect to the epoch number. One epoch corresponds to seven iterations on the training set and three iterations on the validation set. A summary of the training session results is presented in Table 4.2. The best epoch corresponds to the best loss computed on the validation set. The lengthscale, l, and the noise level, σ_n, are two of the key parameters learned by the model.

4.1.3 Data pre-processing

Before training a star cluster model, we adjust the center of mass (CM) position and velocity of the data.

We center the data so that the CM of the star cluster is at $(0, 0, 0)$. This is done easily by subtracting the CM position from the position coordinates (x, y, z) of all stars. The velocity of the CM is actually the bulk motion of the entire cluster. Ideally, the CM velocity should be close to $(0, 0, 0)$, indicating that the cluster is not undergoing significant overall motion. If

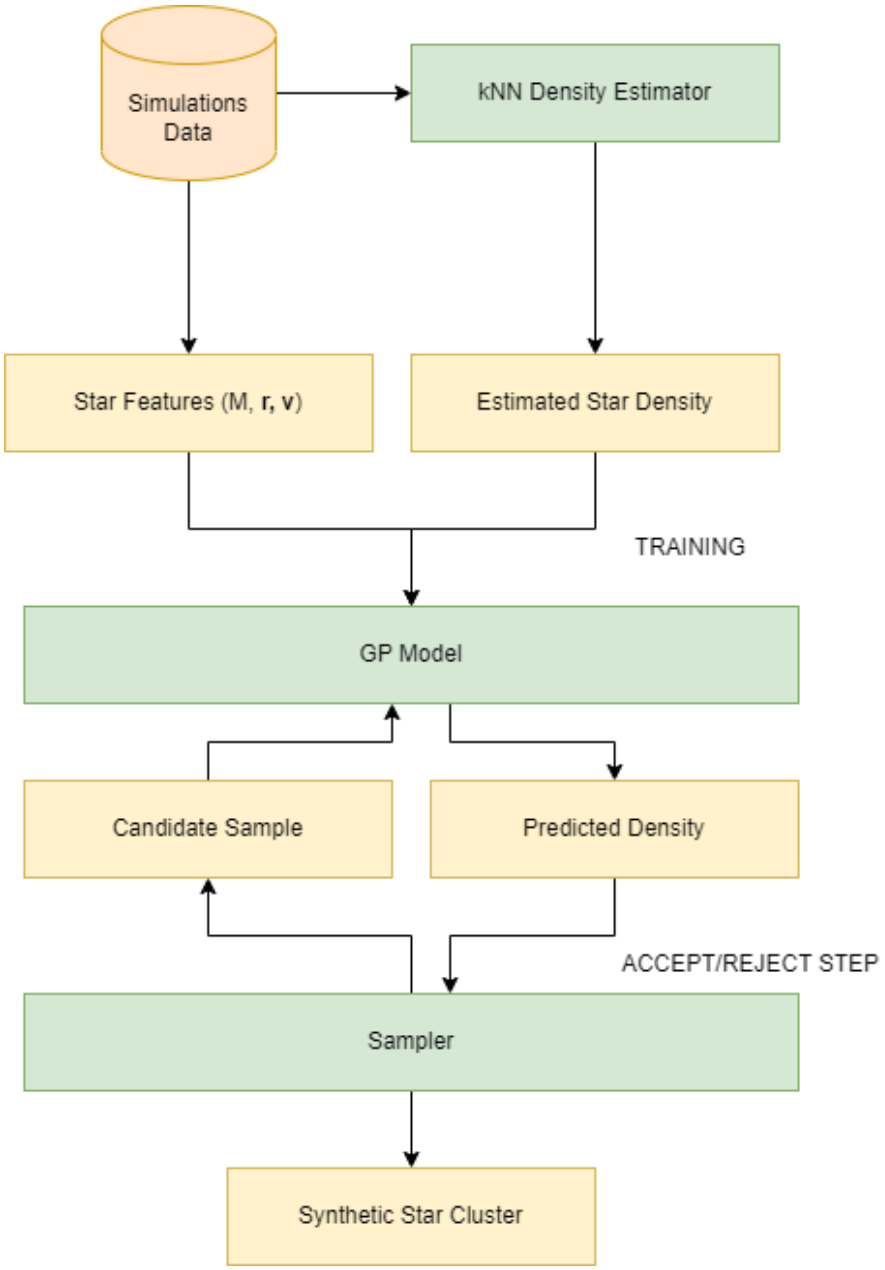


Figure 4.2: Processing pipeline overview

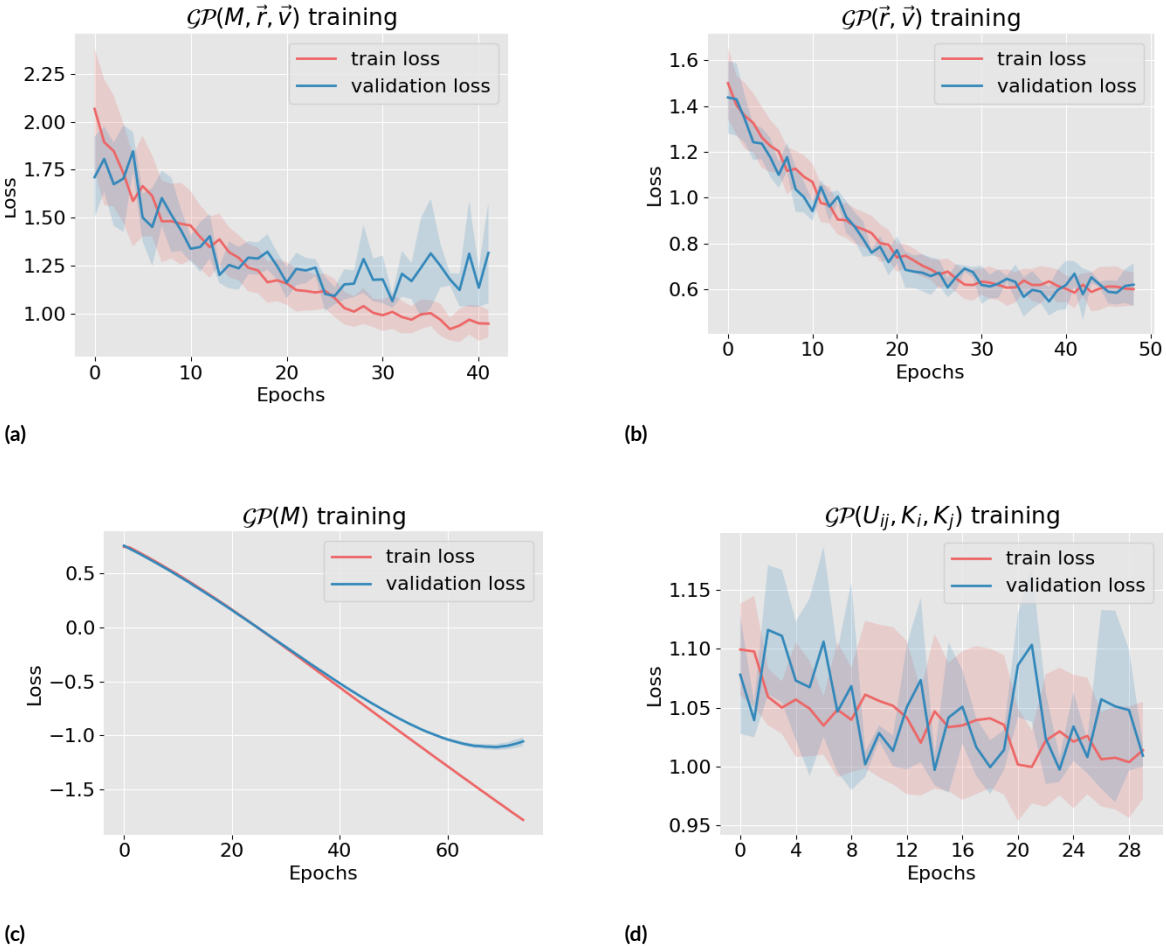


Figure 4.3: Training and validation losses plotted with respect to the epochs. The losses are computed as the mean value of the losses obtained for each cluster of the training/validation set in one epoch. The shadowed areas correspond to the standard deviation.

the CM velocity is significantly non-zero, it could indicate that there is an external influence affecting the cluster. We remove this influence, because it can affect negatively the training process.

By doing these operations, we ensure that the model focuses on the internal structure and we do not affect the cluster dynamics as we just apply some shifts to the data. We avoid $\vec{r}$ and $\vec{v}$ data as it can alter the cluster dynamics. Also, we do not scale the mass values as scaling them back to the initial space is an issue when training on multiple star clusters having different mass distributions.

Logarithmic feature scaling and standardization are performed for the energy values in the training process of $\mathcal{GP}(U_{ij}, K_i, K_j)$. The new energy values become

$$U_{ij}^* = \frac{\log U_{ij} - \mathbb{E}(\log U)}{\sqrt{Var(\log U)}} \qquad K_i^* = \frac{\log K_i - \mathbb{E}(\log K)}{\sqrt{Var(\log K)}} \tag{4.1}$$

We use logarithmic scaling to reduce the skewness of the feature and standardization to improve the training process. We scale them back after the sampling procedure as we use these values in computing other physical values for determining the parameters of the stars we sample.

For each GP model, we estimate the probability density function values using a k-NN density estimator with $k = 50$. After running the estimator on the discrete set of features of one star cluster, the estimated values are normalized using the following procedure,

$$y_i^* = \frac{N y_i}{\sum_{j=1}^{N} y_j} \tag{4.2}$$

The values $\mathbf{y}^*$ will be the target distribution of the GP model.

4.1.4 Sampling and evaluation

We explore several sampling methods including rejection sampling, MCMC methods, and APES for sampling with the 7-dimensional GP model and EMCMC sampling combined with three GP models (energy, mass, and phase-space models).

For each sampling method, we implement the Metropolis acceptance condition. As we do not know where to start the sampling, we use a burn-in [29] of 500 iterations for finding a high density region. The target distribution of the sampling algorithm is given by the density outputs of the GP model.

We do not evaluate the cluster during sampling. We run the algorithm until we sample a

number of N stars. In our experiments, we set $N = 3000$. Therefore, we obtain clusters of 3000 stars. We evaluate these clusters by computing several distributions or macroscopic quantities such as the virial ratio, the total mass of the stars, and the number of binaries.

The virial ratio is a dimensionless quantity that describes the dynamical state of a star cluster. It is defined as $\alpha_{vir} = 2K/W$ ([30]), where K is total kinetic energy (the sum of the kinetic energies of all the stars) and W is the total potential energy of the cluster (given by the sum of the gravitational potential energies of all pairs of stars in the cluster).

The virial ratio is a useful tool for studying the evolution of stellar clusters and can be used to understand their dynamical state and the physical processes that drive their evolution. If $\alpha_{vir} = 1$, then the cluster is in a state of dynamical equilibrium. Otherwise, if it is greater than 1, then the cluster is in a state of expansion, or if it is lower than 1, then there is a contraction state.

The number of binaries is computed as the total number of stars that are bound gravitationally to another star. This is verified by comparing the potential energy between two stars to the sum of their kinetic energies, computed with respect to their center of mass. If $K_1 + K_2 < |U_{12}|$, then the system is bound. By computing this quantity, we do not count separately the number of binary star systems, triple star systems, and so on.

4.2 Sampling from 7D model

4.2.1 Rejection sampling

This method is straightforward and does not need parameter tuning. However, this means that we cannot control in any way the density distribution of the sampled clusters and we notice that its results are very far from what we expect. We have sampled a cluster containing 3000 stars obtaining a virial ratio of 41.4, total mass of 128.3 solar masses and two binaries. Comparing these values to the statistics retrieved from the simulated clusters (Table 4.1), we notice that the clusters sampled via rejection are categorically unreliable.

4.2.2 Metropolis algorithm

Here we analyze the properties of the new generated clusters Monte-Carlo methods based on the Metropolis-Hastings algorithm.

Firstly, we try different step sizes between 0.1 and 1.0 for the MCMC sampling algorithm.

We notice that the step size does not have a visible effect on the acceptance rate. We define the acceptance rate as the ratio between the total number of accepted samples, i.e. the number of stars in the final cluster, N, and the total number of star candidates, $N + R$, where R is the number of the rejected candidates according to the Metropolis-Hastings rule.

We perform another runs trying different variants of the sampling method. We try sampling the step size from a normal distribution, $\mathcal{N}(\mu, \sigma)$ (random walk - RWMC). We set μ to 0.5 and 1.0. The standard deviation σ is set to 0.5. Next, we implement a non-Markovian approach (NMMC), by sampling each time with respect to a randomly sample accepted previously. We try this approach with and without random walk.

The sampling points are bound by the average of the upper and lower limits of the features of each training cluster. For each implemented Monte-Carlo method, the new points are drawn from a truncated normal distribution such that we do not exceed these bounds.

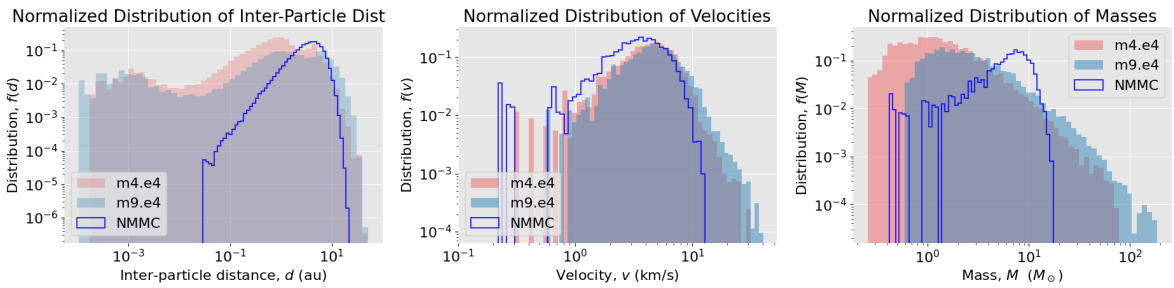
For each sampled cluster, we compute several macroscopic physical quantities. The virial ratio, the number of binaries, and the total mass. In Table 4.3, we present a summary of all of these results, computed for different sampling approaches that use the Metropolis algorithm. One can notice a broad spectrum of results.

The virial ratio shows an increasing trend with respect to the step size for the MCMC algorithm with fixed step size (clusters 1-5), whereas the number of binaries is decreasing. However, we can see that for small step sizes, the number of binaries is clearly too high. If we are comparing to the statistics of the simulated clusters (Table 4.1), cluster 4 has a reasonable number of binaries, but the virial ratio is not suitable. The same is also valid for the clusters sampled with RWMC (6 and 7). However, cluster 9 seems interesting. In this case, the number of binaries is slightly lower than the average from Table 4.1, but the virial ratio and the total mass are comparable. Cluster 8 has been obtained with the same method (NMMC) as cluster 9 but different step-size. Thus, the step size can be a decisive parameter when sampling.

If we observe the in-depth morphology of cluster 9, we see that there is a gap of low inter-particle distances (Figure 4.4 (a)). This gap exists for all the generated clusters in Table 4.3. Thus, all the binaries we count are distant binaries. If the stars forming a binary system are far away from each other, the potential energy is not high enough to keep the system bound because they could be subject to the external forces from the background stars. Considering the dynamical evolution of the cluster, we could expect these pairs to unbound after a certain time or to get closer in distance forming a stable binary system. Therefore, the number of binaries shown in Table 4.3 is an upper limit of the stable binaries. Even if this upper limit is large, the lack of low inter-particle distances can lead to a few stable binaries.

#	Markovian	step size	random walk	acc. rate	α_{vir}	# binaries	M ($10^3 M_\odot$)
1	✓	0.10	✗	0.3312	0.60	2980	13.2
2	✓	0.25	✗	0.3373	5.27	2334	20.7
3	✓	0.50	✗	0.3407	18.2	1679	68.4
4	✓	0.75	✗	0.3306	14.1	441	56.6
5	✓	1.00	✗	0.3419	43.5	361	102.5
6	✓	$\mathcal{N}(0.5, 0.5)$	✓	0.3305	13.2	1111	98.9
7	✓	$\mathcal{N}(1.0, 0.5)$	✓	0.3271	11.1	371	180.2
8	✗	0.50	✗	0.3279	0.09	2554	63.5
9	✗	1.00	✗	0.3320	1.87	216	23.3
10	✗	$\mathcal{N}(0.5, 0.5)$	✓	0.3338	0.32	1730	44.8
11	✗	$\mathcal{N}(1.0, 0.5)$	✓	0.3376	3.61	99	19.8

Table 4.3: Monte-Carlo sampling results using Metropolis algorithm. The generated clusters contain 3000 stars. Columns: (1) Markovian methods, (2) the step size, (3) random walk methods, (4) the acceptation rate, (5) the virial ratio, (6) the number of binaries, (7) the total mass of the sampled cluster.



(a) (b) (c)

Figure 4.4: Distributions of inter-particle distances (a), velocity (b), and mass (c) for the cluster generated by NMMC with step size 1.0 (blue line). Simulated clusters [1] are represented for comparison: m4e4, and m9e4.

We see in Figure 4.4 (b) that the distribution of the velocity magnitudes is comparable to those computed from simulations. However, regarding the mass distribution in Figure 4.4 (c), we see that the there are more massive stars than what we would expect from simulations.

4.2.3 APES ALGORITHM

We implement APES using the *multiprocessing* library of Python [*]. To approximate the distribution $\tilde{\pi}$, we use the same methods as presented in [24]. We implement both Gaussian and Student's t kernels. The weights are updated during each iteration solving the NNLS prob-

[*] `https://docs.python.org/3/library/multiprocessing.html`

algorithm parameters					cluster data			
#	L	N	kernel	acc. rate	# stars	virial ratio	$M\,(M_\odot)$	# binaries
1	50	100	Student	0.5880	3315	5.95	9.6	331
2	60	85	Student	0.7135	3639	5.94	15.2	492
3	80	55	Student	0.6534	2874	6.45	7.9	276
4	100	45	Student	0.6522	2935	4.56	8.4	343
5	80	55	Gauss	0.7411	3261	11.07	20.3	532
6	50	100	Gauss	0.7323	3296	15.75	20.8	536

Table 4.4: Results of APES algorithm runs using different ensemble lengths, L, and kernels. The maximum number of iterations, N, is set such that we sample a number of stars around 3000. For the generated clusters, we determine the virial ratio, total mass, M, and the number of binaries.

lem, $min||Kw - \pi||^2$, subject to $w \geq 0$. For that, we use the *nnls* function implemented by *scipy.optimize*. The full implementation is available on the GitHub repository [†].

Our approach is Markovian, the new candidate samples are drawn from a truncated normal distribution with a standard deviation of 0.5 for position and velocity components, and 1.0 for the mass. The phase space bound is a sphere with the radius of 15 pc. The mass is bound between 0.1 and 100 solar masses. The initial ensemble is drawn from a similar distribution centered on the origin of the phase space. For all the runs, we set a burn in of 25 iterations.

We use ensemble of different lengths from 50 to 100. As it is shown in Table 4.4, the acceptance rate for Student kernel varies between 58% and 71%, whereas the use of a Gaussian kernel leads to higher acceptance rates. The maximum number of iterations, N, varies according to the ensemble length, L, as we aim to sample clusters that contain around 3000 stars. The total number of stars depends on the acceptance rate as we do not know how many walkers will accept the new candidates. Certainly, one can modify the algorithm to discard the extra samples accepted by the ensembles, but it involves the risk of losing correlated samples.

We see that the clusters created via Gaussian kernels are hyper-kinetic with an unrealistic virial ratio around 10-15. One might observe that the Student kernel yields lower virial ratios, suggesting clusters that are closer to a bound state. On the other hand, taking a look over the distributions of the physical quantities, we notice that the gap problem persists in the case of the low inter-particle distances. In Figure 4.5 we plot these distributions for the cluster 4 from Table 4.4. We observe that APES samples better mass distributions with respect to the Metropolis algorithm.

[†] `https://github.com/prodangp/star-clusters-gen`

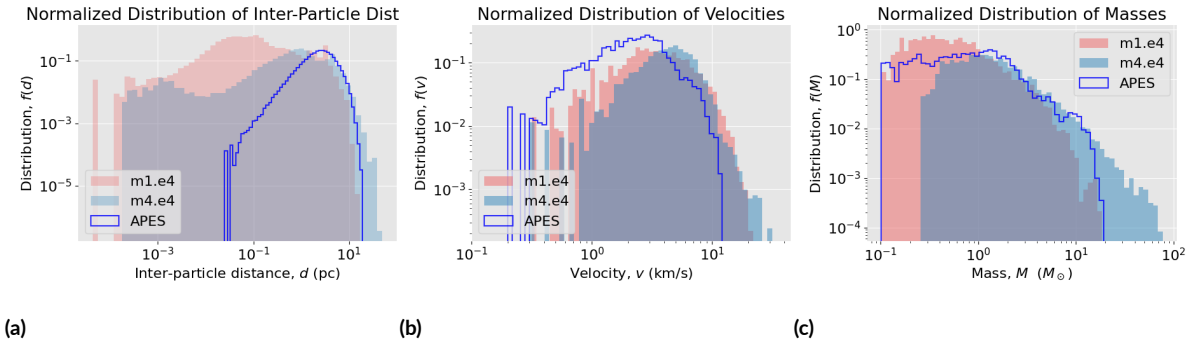


Figure 4.5: Distributions of inter-particle distances (a), velocity (b), and mass (c) for a cluster with 2935 stars generated by APES with an ensemble of length 100 and Student kernel. Simulated clusters [1] are represented for comparison: m1e4, and m4e4.

4.3 Energy based sampling

We test the sampling algorithm presented in Chapter 3 (EMCMC) using the GP models trained on the clusters generated from the hydrodynamical simulations performed by Ballone et al [1]. The energy states and the masses are sampled using Metropolis algorithm. Firstly, we fine-tune the algorithm parameters. We continue on running several experiments and analyze the generated clusters. We compare the performances of EMCMC to the other methods presented in the previous section.

4.3.1 Hyper-parameters tuning

We start by fine-tuning the EMCMC hyperparamenters: the step-size, s, the maximum jump between two consecutive sampled stars in the position space J, and the number of direction proposals, N_c, when sampling the candidates in the phase-space. We have observed that a step-size between 0.2 and 0.4 leads to an optimal exploration of the energy space.

For different values of J, we run the algorithm without inserting the noise generated by the likelihood function of the GP model. The probability distributions are given by the posterior mean calculated using the GP models. In this way, we have a better focus on the effect produced by the hyperparameter, J. We show in Figure 4.6, the virial ratio and the upper limit of the stable binaries with respect to the maximum jump size. One can notice an ascending trend for the virial ratio and a descending one for the number of binaries. As expected, constraining to smaller jump sizes results in clusters that are more tightly bound and have a greater number of binaries. The acceptance rate slightly varies between 70% and 75%.

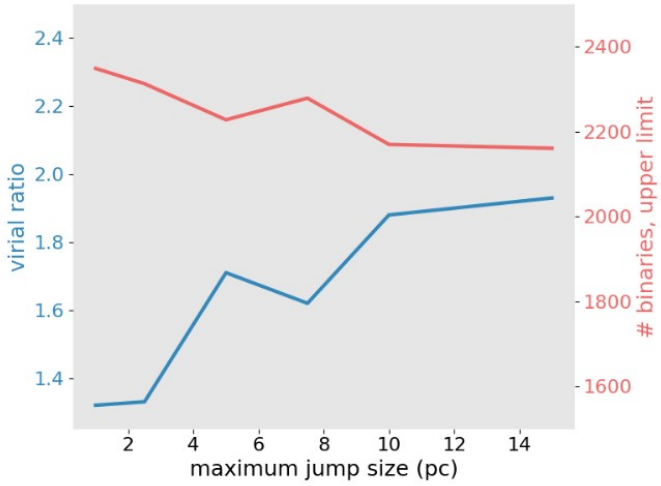


Figure 4.6: The effects of the maximum jump size allowed during EMCMC sampling ($s = 0.4, N_c = 1000$) on the generated clusters. The virial ratio (blue, left axis) and the number of binaries (red, right axis) are plotted with respect to the maximum jump size.

Varying the parameter N_c, we have observed that the optimal value is around 1000. We have noticed that smaller values around 100 affect the results negatively, because the exploration of the phase-space is limited. In contrast, better exploration with $N = 5000$ leads to similar results, but longer runs as the GP model has to compute more predictions. Taking into account these aspects, we consider that $N_c = 1000$ is optimal in terms of exploration capability and time consumption.

4.3.2 Newly generated clusters

We continue our EMCMC experiments inserting the noise in the predictions of the GP models. Ten clusters are sampled using the $N_c = 1000$, small J values, and step size of 0.2 or 0.4. The results are presented in Table 4.5. With respect to the previous runs, we observe a drop in the acceptance rate by $20 - 30\%$ due to the noisy predictions. Most of the generated clusters are close to stability with virial ratios between 1 and 2. The total mass depends on the constraint we have used for the mass distribution. The clusters 1-5 have a lower bound for the masses of $0.1M_\odot$. For the other clusters the lower bound is $0.5M_\odot$, but different sampling step sizes have been used to obtain diverse distributions.

The number of binaries is high for every cluster in Table 4.5. However, the inter-particle distance distribution does not have any gap with respect to the simulations data 4.7. This means

algorithm parameters				cluster data		
#	J (pc)	s	acc. rate	virial ratio	$M\,(10^3 M_\odot)$	# binaries
1	0.75	0.4	0.3925	2.73	7.2	1739
2	1	0.2	0.5691	1.57	7.2	2078
3	1.25	0.4	0.4105	2.8	7.2	1757
4	1.5	0.2	0.5667	1.69	7.2	2089
5	2	0.2	0.4663	4.16	7.2	1721
6	0.75	0.4	0.4652	2.78	6.6	2012
7	1	0.2	0.4419	1.82	7.6	2204
8	1.25	0.2	0.5285	1.07	11.6	2359
9	1.5	0.2	0.5097	1.08	13.5	1334
10	1	0.4	0.3534	1.21	32.2	1401

Table 4.5: Results of EMCMC Metropolis runs using different sets of hyper-parameters: J, s and N_c set to 1000. We show the acceptance rate of the algorithm for $N = 3000$ stars. For the generated clusters, we determine the virial ratio, total mass, M, and the number of binaries.

that the chance is high that some of the counted binaries are also stable. Further simulations are needed to confirm this.

4.3.3 Comparison to other methods

The advantage of EMCMC algorithm is the capability of sampling clusters exhibiting an appropriate spectrum of inter-particle distances. As we have seen previously, sampling directly in the 7-dimensional feature space of the stars leads to improper inter-particle distance distributions. However, we have seen that in APES algorithm, the virial ratio and the inter-particle distance distribution of the generated clusters are influenced by the kernel. There is a possiblity of finding the right kernel for this sampling problem that can lead to better results in the case of APES. At the moment, we have not made any attempt to find such a kernel.

Regarding the virial ratio, rejection and APES generate realizations with an unusual dynamic having virial ratios of above 5. Metropolis algorithm provides clusters having a variety of dynamical states, with one out of ten clusters having a realistic virial ratio. The only method that is consistent in generating cluters having realistic virial ratios is EMCMC Metropolis. From 10 generated clusters, 6 have a virial ratio in the interval [1, 2]. Also, the other virial ratios are up to a value of 4.

All of the experiments are performed for generating clusters with a number of 3000 stars. Therefore, it is possible to make a comparison of the computational performances. Table 4.6

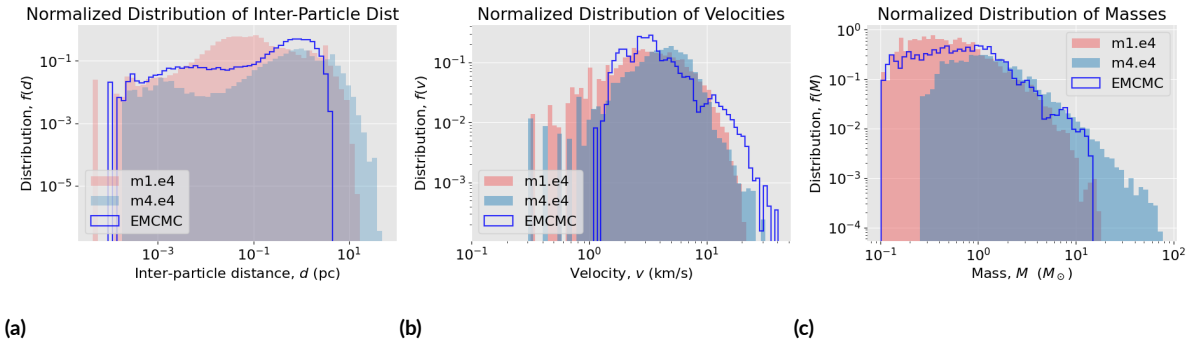


Figure 4.7: Distributions of inter-particle distances (a), velocity (b), and mass (c) for a cluster with 3000 stars generated by EMCMC with a step size of, $s = 0.2$, maximum jump size, $J = 1.5$, and $N_c = 1000$. The mass distribution has a lower bound of 0.1. Simulated clusters [1] are represented for comparison: m1e4, and m4e4.

Algorithm	Runtime (min)	Memory GPU (GB)	Memory CPU (GB)
rejection	7.5 ± 1.0	1.0	1.1
Metropolis	6.5 ± 1.5	1.0	1.1
APES (3 workers)	14.5 ± 2.5		1.2
APES (4 workers)	10.5 ± 1.5		1.4
APES (5 workers)	9.0 ± 1.0		1.6
EMCMC ($N_c = 200$)	8.5 ± 1.5	1.1	1.0
EMCMC ($N_c = 1000$)	16.0 ± 3.5	1.2	1.0
EMCMC ($N_c = 5000$)	47.0 ± 6.5	1.9	1.0

Table 4.6: Time and memory analysis of the sampling algorithms. There are provided the average values and standard deviations of the runtime, the GPU memory usage and CPU memory usage. No GPU memory values are shown for APES, because the GPU is not used in that case. We show the runtime and CPU memory usage for different number of workers used in the parallelized loops of APES. For EMCMC, we present the results for different values of N_c parameter.

shows a summary of the time and memory analysis of the algorithms. It seems that Metropolis algorithm is the fastest option, while APES uses multiprocessing reducing the runtime, without consuming at all GPU resources. The GPU used is NVIDIA GeForce RTX 3060, and the CPU is provided by a 12th Gen Intel Core i7-12700H with 4.70 GHz frequency.

In Figure 4.8 we show the density distribution in xy-plane of coordinates. We observe that MCMC sampling generates distributions that are focused in several branches having different geometric properties with respect to the simulations. NMMC and APES provide more spherical distributions, but the spatial density of the stars is lower than the one of the simulated clusters. On the other hand, the resemblance of EMCMC-generated clusters to the training set stands out (last three subfigures of Fig. 4.8). Visually, as Figure 4.8 elucidates, the disparity in the spatial density and geometry between the methods is evident.

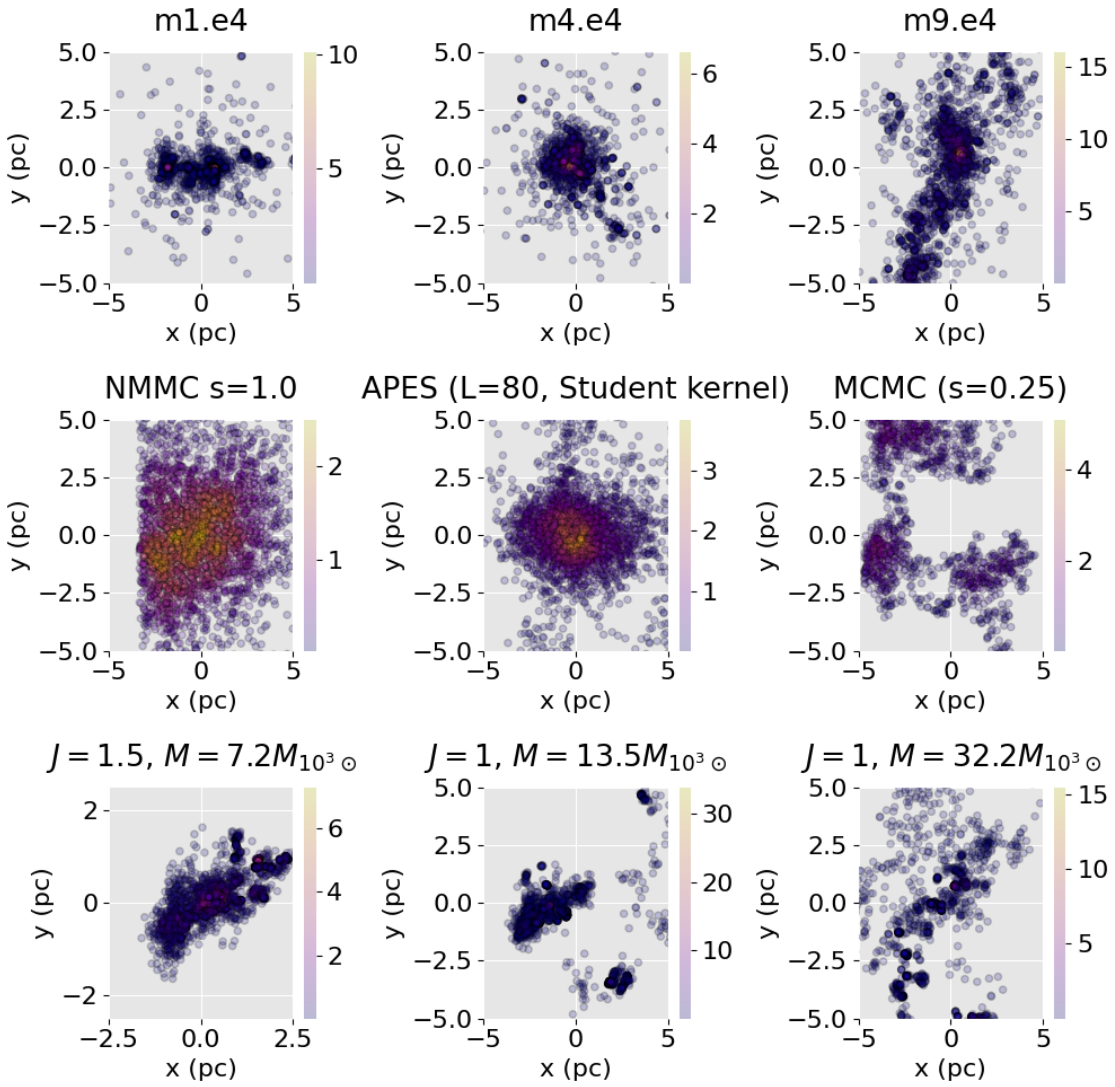


Figure 4.8: Density distribution in the xy-plane of coordinates for various sampling methods. The first three subfigures correspond to simulated clusters from [1]. Next three subfigures are for Monte-Carlo generated clusters, the step-size, s, is shown for NMMC and MCMC methods, and for APES the ensemble length, L. The MCMC realization demonstrate distributions with distinct geometric branches, while NMMC and APES depict more spherical distributions with sparser spatial densities. In contrast, the geometry and spatial density of EMCMC closely align with the properties of the training dataset clusters, as showcased in the last three subfigures, where J is the maximum jump size when sampling and M the total mass of the emulated cluster.

5
Conclusion

The learning framework we propose, GP modeling of star clusters and sampling from the target distribution for generating new clusters, is a two-step process that involves building a statistical model of the data using Gaussian processes (GPs) and then using this model to generate new samples from the underlying distribution. We saw that the process of sampling from the target distribution is indeed challenging and careful exploration is needed to fine-tune the algorithms used.

The GP model was trained on the probability density function of the stars in the cluster, and different sampling methods such as Monte Carlo methods or APES were used to generate new star cluster realizations. In the endeavor to sample clusters with distinctive spatial distributions and dynamical states, the EMCMC algorithm has presented itself as a particularly adept solution, especially when considering the critical spectrum of inter-particle distances. Direct sampling in the expansive 7-dimensional space can, without the right precautions, yield undesired distance distributions—a pitfall that certain methods like APES might avoid with optimal kernel choices, though such kernels remain to be identified.

While most traditional sampling methods largely operate on a mathematical or heuristic basis, EMCMC goes a step further by leveraging the underlying physical principles that dictate the behavior and properties of star clusters. This inclusion of physics ensures that the generated samples are not just mathematically plausible, but also physically meaningful. It is this infusion of real-world principles that contributes to the EMCMC's consistent ability to generate clusters with realistic virial ratios and spatial distributions.

One potential alternative to this two-step process is to use a single deep learning model to model the data distribution and generate new samples. This way we can also learn the parameters involved in sampling new clusters without the need of exploring different sampling approaches. Deep learning models, such as generative adversarial networks (GANs) [31] or variational autoencoders (VAEs) [32], can learn complex and high-dimensional distributions directly from data. While a single deep learning model could replace the two-step process of GP modeling and sampling from the target distribution for generating new star clusters, it is important to carefully consider the limitations and potential trade-offs of this approach as well.

Another difficulty we found is how to validate the model's performance or assess its accuracy, as there may be no explicit mathematical representation of the data distribution. This also would apply for deep learning models.

Overall, the use of machine learning techniques and statistical models such as GPs can significantly reduce the computational costs and time needed to generate diverse sets of initial conditions for N-body simulations, providing a valuable tool for astronomers in this field of study.

References

[1] A. Ballone, M. Mapelli, U. N. D. Carlo, S. Torniamenti, M. Spera, and S. Rastello, "Evolution of fractality and rotation in embedded star clusters," *Monthly Notices of the Royal Astronomical Society*, vol. 496, no. 1, pp. 49–59, jun 2020. [Online]. Available: https://doi.org/10.1093%2Fmnras%2Fstaa1383

[2] M. R. Krumholz, C. F. McKee, and J. Bland-Hawthorn, "Star clusters across cosmic time," *Annual Review of Astronomy and Astrophysics*, vol. 57, no. 1, pp. 227–303, 2019. [Online]. Available: https://doi.org/10.1146/annurev-astro-091918-104430

[3] M. G. H. Krause, S. S. R. Offner, C. Charbonnel, M. Gieles, R. S. Klessen, E. Vázquez-Semadeni, J. Ballesteros-Paredes, P. Girichidis, J. M. D. Kruijssen, J. L. Ward, and H. Zinnecker, "The physics of star cluster formation and evolution," *Space Science Reviews*, vol. 216, no. 4, jun 2020. [Online]. Available: https://doi.org/10.1007%2Fs11214-020-00689-4

[4] M. R. Bate, "The importance of radiative feedback for the stellar initial mass function," , vol. 392, no. 4, pp. 1363–1380, Feb. 2009.

[5] S. Torniamenti, "A novel generative method for star clusters from hydrodynamical simulations," *Proceedings of the International Astronomical Union*, vol. 16, no. S362, pp. 141–147, jun 2020. [Online]. Available: https://doi.org/10.1017%2Fs1743921322001703

[6] S. Torniamenti, M. Pasquato, P. D. Cintio, A. Ballone, G. Iorio, M. C. Artale, and M. Mapelli, "Hierarchical generative models for star clusters from hydrodynamical simulations," *Monthly Notices of the Royal Astronomical Society*, vol. 510, no. 2, pp. 2097–2110, dec 2021. [Online]. Available: https://doi.org/10.1093%2Fmnras%2Fstab3608

[7] R. Eckhardt, "Stan ulam, john von neumann, and the monte carlo method," *Los Alamos Science*, vol. 15, pp. 131–136, 1987.

[8] C. Robert and G. Casella, "A short history of markov chain monte carlo: Subjective recollections from incomplete data," *Statistical Science*, vol. 26, no. 1, feb 2011. [Online]. Available: https://doi.org/10.1214%2F10-sts351

[9] C. J. Lada and E. A. Lada, "Embedded clusters in molecular clouds," *Annual Review of Astronomy and Astrophysics*, vol. 41, no. 1, pp. 57–115, sep 2003. [Online]. Available: https://doi.org/10.1146%2Fannurev.astro.41.011802.094844

[10] A. Cartwright and A. P. Whitworth, "The statistical analysis of star clusters," *Monthly Notices of the Royal Astronomical Society*, vol. 348, no. 2, pp. 589–598, 02 2004. [Online]. Available: https://doi.org/10.1111/j.1365-2966.2004.07360.x

[11] R. B. Larson, "Star formation in groups," *Monthly Notices of the Royal Astronomical Society*, vol. 272, no. 1, pp. 213–220, 01 1995. [Online]. Available: https://doi.org/10.1093/mnras/272.1.213

[12] M. Simon, "Clustering of Young Stars in Taurus, Ophiuchus, and the Orion Trapezium," , vol. 482, no. 1, pp. L81–L84, Jun. 1997.

[13] R. J. Parker and M. R. Meyer, "Characterizing the dynamical state of star clusters from snapshots of their spatial distributions," *Monthly Notices of the Royal Astronomical Society*, vol. 427, no. 1, pp. 637–650, nov 2012. [Online]. Available: https://doi.org/10.1111%2Fj.1365-2966.2012.21851.x

[14] G. Iorio, M. Mapelli, G. Costa, M. Spera, G. J. Escobar, C. Sgalletta, A. A. Trani, E. Korb, F. Santoliquido, M. Dall'Amico, N. Gaspari, and A. Bressan, "Compact object mergers: exploring uncertainties from stellar and binary evolution with scpsevn/scp," *Monthly Notices of the Royal Astronomical Society*, vol. 524, no. 1, pp. 426–470, jun 2023. [Online]. Available: https://doi.org/10.1093%2Fmnras%2Fstad1630

[15] I. A. Bonnell, M. R. Bate, and S. G. Vine, "The hierarchical formation of a stellar cluster," *Monthly Notices of the Royal Astronomical Society*, vol. 343, no. 2, pp. 413–418, 08 2003. [Online]. Available: https://doi.org/10.1046/j.1365-8711.2003.06687.x

[16] T. Maschberger, C. J. Clarke, I. A. Bonnell, and P. Kroupa, "Properties of hierarchically forming star clusters," *Monthly Notices of the Royal Astronomical Society*, vol. 404, no. 2, pp. 1061–1080, 2010.

[17] M. Mapelli, "Rotation in young massive star clusters," *Monthly Notices of the Royal Astronomical Society*, vol. 467, no. 3, pp. 3255–3267, 02 2017. [Online]. Available: https://doi.org/10.1093/mnras/stx304

[18] Lee, Yueh-Ning and Hennebelle, Patrick, "Formation of a protocluster: A virialized structure from gravoturbulent collapse - i. simulation of cluster formation in a collapsing molecular cloud," *A&A*, vol. 591, p. A30, 2016. [Online]. Available: https://doi.org/10.1051/0004-6361/201527981

[19] J. Wang, "An intuitive tutorial to gaussian processes regression," 2022.

[20] R.-R. Griffiths, "Applications of gaussian processes at extreme lengthscales: From molecules to black holes," 2023. [Online]. Available: https://www.repository.cam.ac.uk/handle/1810/346223

[21] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning.*, ser. Adaptive computation and machine learning. MIT Press, 2006.

[22] H. Mohammadi, R. Le Riche, and E. Touboul, "A detailed analysis of kernel parameters in Gaussian process-based optimization," 2015. [Online]. Available: https://hal.science/hal-01246677

[23] Y. P. Mack and M. Rosenblatt, "Multivariate k-nearest neighbor density estimates," *Journal of Multivariate Analysis*, vol. 9, pp. 1–15, 1979.

[24] S. D. P. V. S and E. J. Barroso, "APES: Approximate posterior ensemble sampler," *Monthly Notices of the Royal Astronomical Society*, jul 2023. [Online]. Available: https://doi.org/10.1093%2Fmnras%2Fstad2245

[25] L. Tierney, "Markov Chains for Exploring Posterior Distributions," *The Annals of Statistics*, vol. 22, no. 4, pp. 1701 – 1728, 1994. [Online]. Available: https://doi.org/10.1214/aos/1176325750

[26] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," 2018. [Online]. Available: https://arxiv.org/abs/1809.11165

[27] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[28] L. Prechelt, "Early stopping-but when?" in *Neural Networks*, 1996. [Online]. Available: https://api.semanticscholar.org/CorpusID:14049040

[29] V. Roy, "Convergence diagnostics for markov chain monte carlo," *Annual Review of Statistics and Its Application*, vol. 7, no. 1, pp. 387–412, 2020. [Online]. Available: https://doi.org/10.1146/annurev-statistics-031219-041300

[30] M. Mapelli, "Computational astrophysics lectures," 2022.

[31] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[32] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022.