Introduction

The cartpole system is a good introduction to reinforcement learning and model predictive control.

First, the system's trajectories and time evolutions are characterised. Simulated observation data is then gathered and used to fit a linear model and nonlinear models, and their performance evaluated.

A linear policy is formulated to maintain the system in an upright target position, and its robustness characterised. Noise is then introduced and its effects on previous tasks is analysed.

Finally, attempts are made to fit a nonlinear model to the data, which are unfortunately ultimately unsuccessful.

General considerations

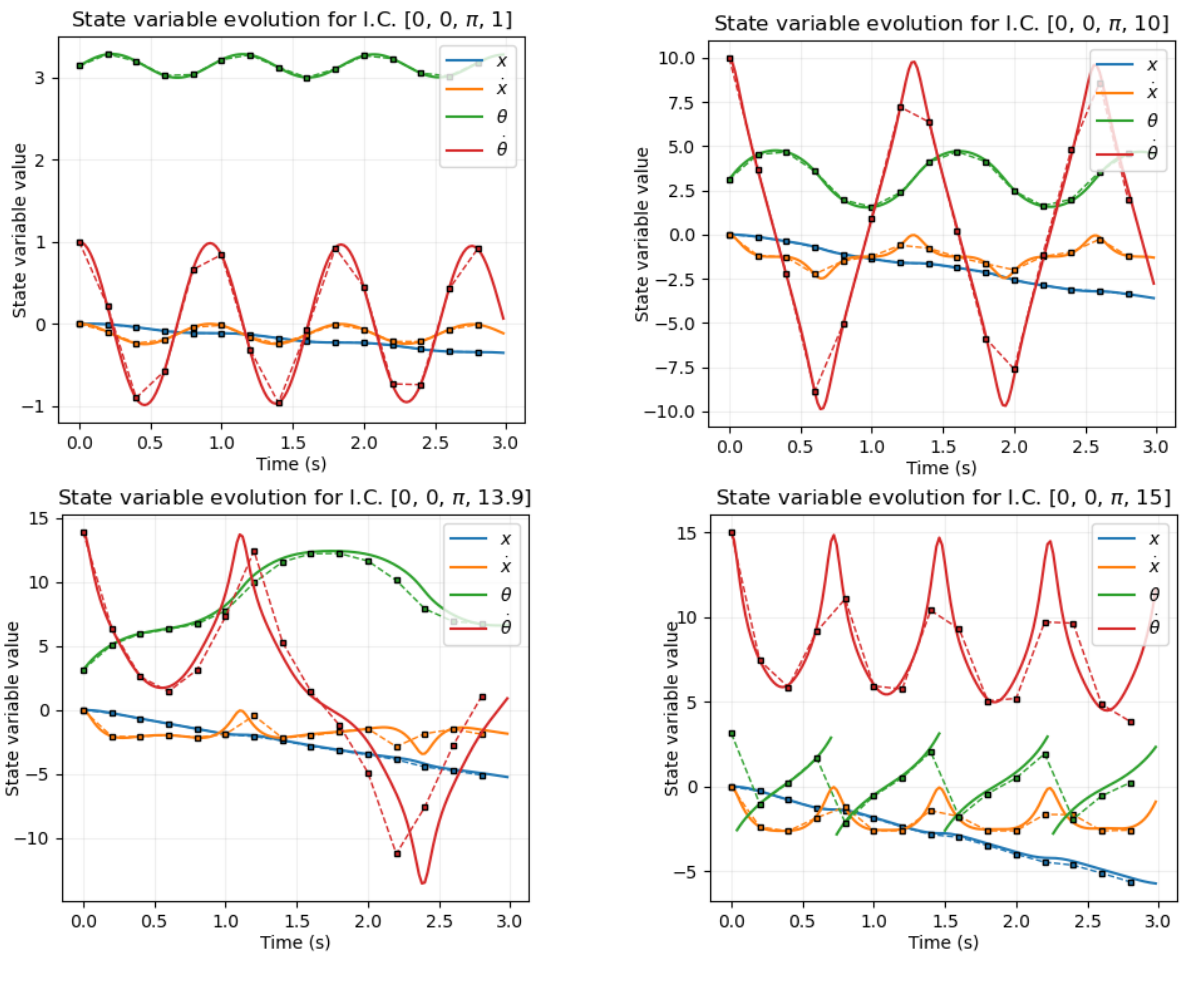
The model runs 50 Euler steps over a time step of 0.2 seconds. The system has oscillating variables, and so the Nyquist criterion should be considered: if the system is observed at less than half the sampling rate of its maximum oscillation frequency, aliasing artefacts will occur. Such trajectories would be sharply nonlinear, and so would be difficult to predict with any model, linear or nonlinear. Since the sampling rate is fixed, it is therefore best to stick to low frequencies, and assume that useful controllers will keep the system state within this domain.

For the pole angular velocity:

$$f_s = 5Hz;\ f_\theta = \dot{\theta}/2\pi \quad \Longrightarrow \dot{\theta} < 5\pi \approx 15.7$$

So $\dot{\theta} \in [-15, 15]$ should be satisfactory. As advised, a corresponding range of $\dot{x} \in [-10, 10]$ will be taken. θ is an angle (centred at the unstable equilibrium) and so $\dot{\theta} \in [-\pi, \pi]$. The system dynamics are invariant to shifts in

Fig 1. Time evolution of system variables with no applied force, starting at the stable equilibrium ($\theta = \pi$), with varying angular velocities. **Solid lines**: Illustrative evolution using a timestep of 0.02 s. **Dashed lines & markers**: Sampled values with the actual timestep (0.2 s). Note that the nature of the Euler scheme causes these pairs to diverge when friction is significant.



x, so x can be set initially to zero for all trajectories, but its modelling range is considered as $[-15, 15]$. Forces F are applied to the system, which are windowed (with a tanh function) to the range $[-20, 20]$, and so the modelled range was $[-25, 25]$ for pre-windowed forces.

Rollouts

Time-trajectories

There are two principal trajectories of the system: *bound* and *unbound*. Bound trajectories occur when the pole oscillates with insufficient energy to reach $\theta = 0$. In the bound, low-energy limit (fig 1a), the pole oscillates sinusoidally about $\theta = \pi$. The cart's velocity couples to this, oscillating in phase with the $\dot{\theta}$ but with a much smaller amplitude. However, it remains of an opposite sign to the initial angular velocity, so that the cart position "drifts" monotonically. Conversely, unbound trajectories have sufficiently high energy that the pole repeatedly "flips", causing θ to monotonically increase (sans remapping) (fig 1d). Between these extremes, the behaviour stays broadly similar either side of the critical transition energy ($E = 0$, defining zero potential at $\theta = 0$ [A1]), and the monotonic cart "drifting" remains, though the waveforms change shape in a nontrivial way (fig 1b). In all cases, friction gradually causes the energy to diminish and the oscillation amplitudes to reduce, so that bound states decay to zero and unbound states eventually become bound (fig 1c). This effect of friction can be largely decoupled from the broader system dynamics, especially when a controller is present to offset the energy loss.

Adding a nonzero cart velocity leaves the pole dynamics mostly unchanged (see next section), and shifts in initial cart position are trivial.

Phase portraits

Since the cart position is trivial to deduce given its velocity history, and the timescale of energy loss due to friction is much greater than the timescale of the cart dynamics, a qualitative understanding of the state-space trajectories of this system can be gained entirely via energy embedded in the 3d space formed by $[\dot{x}, \theta, \dot{\theta}]$.

This consists of ellipses in the $\dot{x}$-$\dot{\theta}$ plane modulated in size and tilt angle by the value of θ. When $E > 0$, a continuous, periodic tube is formed. It splits into disjoint surfaces at $E = 0$, which shrink into ellipsoids as E reduces to its minimal value. Over longer time periods, trajectories migrate onto lower-energy

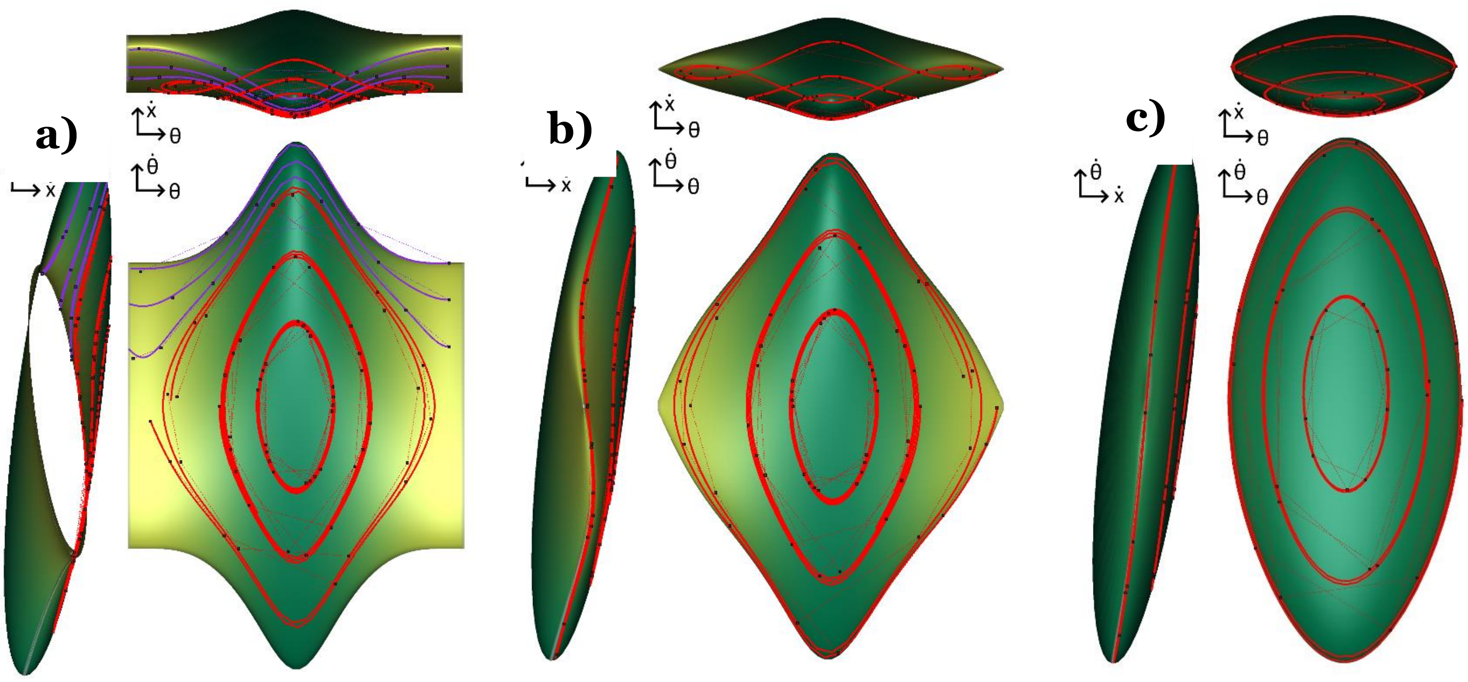


Fig 2. Orthographic views of trajectories in $[\dot{x}, \theta, \dot{\theta}]$-space with initial energies equal to 1, 0, and -2 for **a), b), c)** respectively, superimposed onto energy isosurfaces at those values. **Red lines**: Bound trajectories. **Purple lines:** Unbound. **Solid lines:** Illustrative, with 0.02 s timestep. **Dotted lines & markers:** 0.2 s timestep.

isosurfaces until they become bound and oscillations decay.

Changes of State

Formulation

Suppose $\boldsymbol{X}$ is the current state of the system, including the force F as a 5th component. After one step of time evolution (one "action"), let $\boldsymbol{Z}$ be the next state, and define $\boldsymbol{Y} = \boldsymbol{Z} - \boldsymbol{X}$. The goal of modelling this system is to learn the mapping $\boldsymbol{g}: \mathbb{R}^5 \to \mathbb{R}^4, \boldsymbol{g}(\boldsymbol{X}) = \boldsymbol{Z}$, or equivalently $\boldsymbol{f}: \mathbb{R}^5 \to \mathbb{R}^4, \boldsymbol{f}(\boldsymbol{X}) = \boldsymbol{Y}$.

General Properties of $\boldsymbol{f}$ and $\boldsymbol{g}$

Over the relatively small timestep T=0.2 s, variables should generally evolve in a relatively simple way. In fact, many pairs of input-output variables have approximately linear relationships for $\boldsymbol{g}(\boldsymbol{X})$. This is demonstrable by initialising the system in random states, scanning alternately over the full ranges of state variables, and calculating correlation coefficients with the output state variables. Figure 3a displays the mean of such values over 100 initial states – clearly a number of matrix elements imply linearity, at least for particular settings of other parameters. In particular, since each variable does not change much in a timestep, it is correlated strongly to itself, and the gradient of its best fit line is approximately equal to 1 (fig. 3b).

Subtracting the initial state gives $\boldsymbol{f}(\boldsymbol{X})$ and removes these diagonal correlations – though the others remain (fig. 3c,d)– as well as removing constant terms. If we fit a linear model to $\boldsymbol{f}(\boldsymbol{X})$, we can hope to capture these particular relationships, hoping that the gradients do not change significantly over the range of other parameter settings.

The nonlinear parameter dependencies of $\boldsymbol{f}$ which have been obscured by looking for linear relationships can be approximately found by evaluating the gradients at random points, and summing their absolute values over many samples (fig 3e). With this in mind, we can entirely exclude dependence on x, and establish that dependence on $\dot{x}$ is indeed weak except of course for x. However, a better

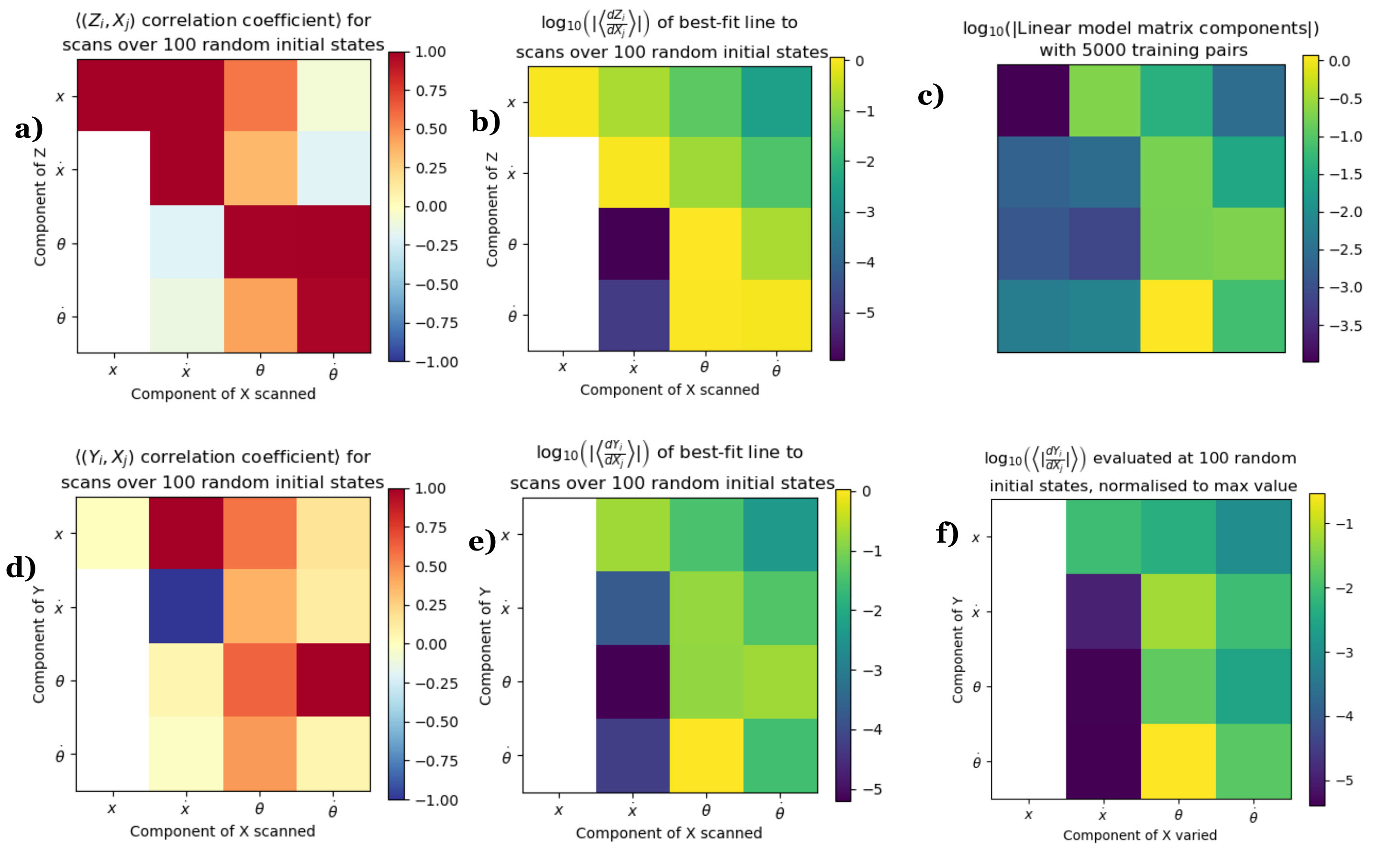


Fig 3. Colour maps of: **a), d)** Mean Pearson correlation coefficient between change in function value and initial state scanned over full range for 100 random initial states for $\boldsymbol{g}$ and $\boldsymbol{f}$; **b), e)** Log magnitude of mean fitted gradient for corresponding scans; **c)** Components of the linear model matrix fit to 5000 random (Y, X) data pairs; **f)** Log mean magnitudes of gradient components evaluated at 100 random initial states. *White-coloured values are those with magnitudes < 1e-12, and so considered to be zero, or fully correlated. F is excluded.*

picture of nonlinear mappings should be obtained first.

Single variable scans

These results about linearity can be reaffirmed visually by taking random start states, and plotting how those states vary as a single variable is scanned through (fig. 4).

Understanding in Physical Terms

Each state variable's evolution depends fairly strongly on θ and $\dot{\theta}$. Variables only have a very weak dependence on $\dot{x}$, except of course x. The variables are all completely independent of x.

These traits can be understood by considering the dynamical equations of the system [A2], where causal connections for an infinitesimal timestep are depicted schematically in fig. 5. There is a general flow of information downward through the figure, with $\dot{x}$ only affecting the angular variables and itself through the weak effects of friction. During the 50 Euler iterations per timestep, the information flow is diffused through the network, resulting in the causality matrix of Figure 3e. In this sense, the angular variables completely drive the dynamics, and so it is important to take care to model them well.

Nonlinear Properties of $\boldsymbol{f}$

Although friction is important on longer time scales, the goal is to actively control the system, and so the shorter-term dynamics are the only important aspects to model. Therefore all of the components of $\boldsymbol{f}(\boldsymbol{X})$ except the first can be characterised in a contour plot against θ and $\dot{\theta}$ (ignoring the dependence on F). For the evolution of cart position, the relationship with $\dot{x}$ will be simply $\frac{df_1}{d\dot{x}} = 0.2\dot{x}$ excluding friction, so with this in mind, a contour plot against θ and $\dot{\theta}$ with $\dot{x} = 0$ is again sufficient for understanding. These plots are shown in fig. 6. It is clear that for these mappings, a linear function will be insufficient to approximate the entire landscape. For the evolution of θ, the broad behaviour will be correct (ignoring the "wiggles"), and for the evolution of $\dot{\theta}$, a linear model trained from data in the central diagonal region would roughly characterise the local contours. However, the full range of θ needs to be modelled, so a linear model

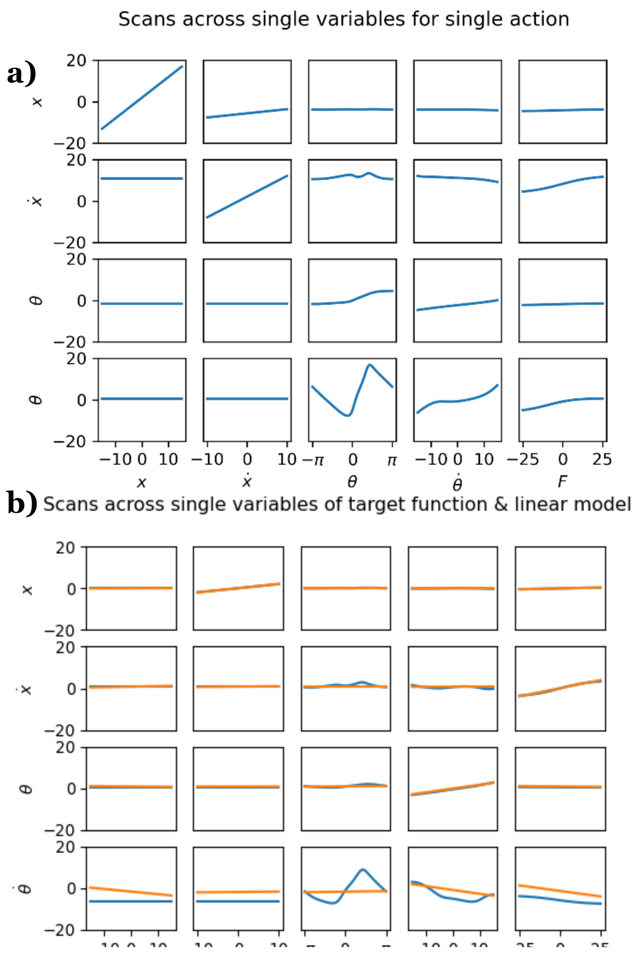
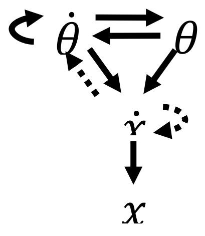


Fig 4. Single variables scans, **a)** for a single action, **b)** for the target function (blue) and linear model (orange).

cannot be expected to model the oscillatory relationship perpendicular to these contours well. For x and $\dot{x}$, the function is near zero in the majority of the landscape – which can be reproduced trivially with a linear model – but there are hills and basins at large $\dot{\theta}$, which definitely cannot. However, if $\dot{\theta}$ is kept sufficiently small, an approximately zero



$$\begin{tikzcd}
\dot{\theta} \arrow[loop left] \arrow[r, shift left] \arrow[dr] & & \theta \arrow[l, shift left] \arrow[dl] \\
& \dot{x} \arrow[ul, dashed] \arrow[loop right, dashed] \arrow[d] & \\
& x &
\end{tikzcd}$$

Fig 5. Lines represent the flow of information and causal changes between state variables over an infinitesimally short timestep. Solid lines indicate that the dependence is important on short timescales; dashed lines indicate that the only dependence is via the weak effects of friction. Force affects $\dot{\theta}$ and $\dot{x}$.

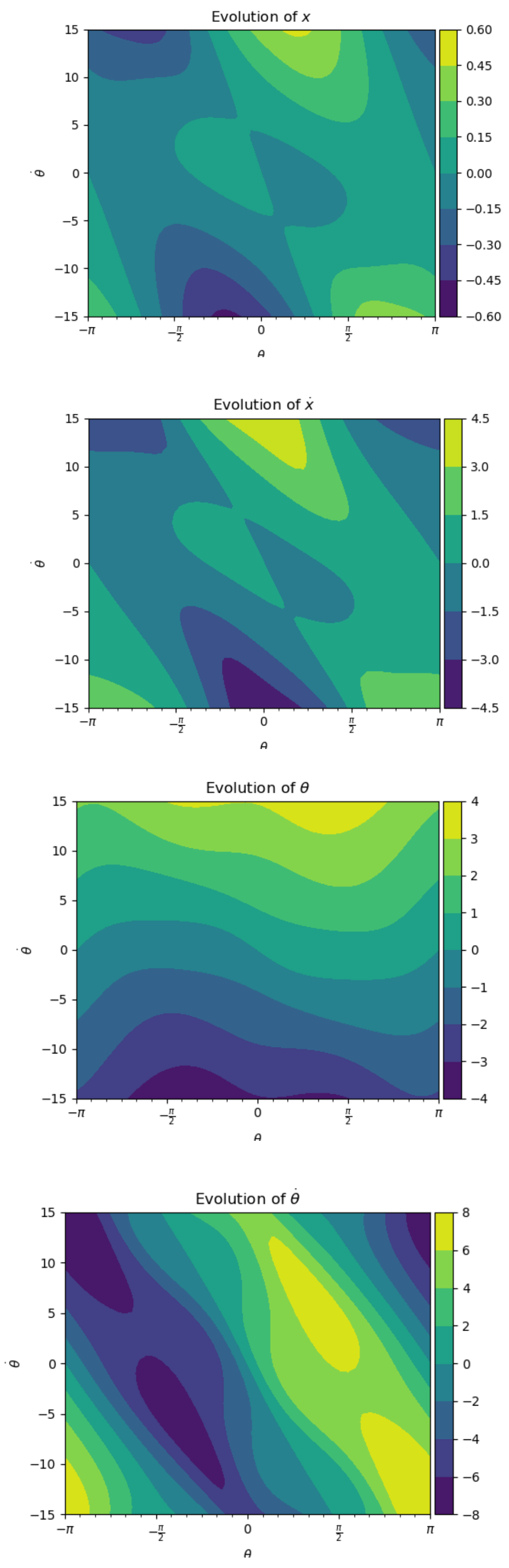


Fig 6. Contour plots of state variable evolutions as functions of θ and $\dot{\theta}$.

function could work moderately well, excluding the shallower central wavelet-like variation which corresponds to the minor couple oscillations in fig. 1.

Linear model

Formulation

For each component Y_i of $\boldsymbol{Y}$, assume the dependence on $\boldsymbol{X}$ is linear. As discussed, the constant term is then zero, and for each Y_i,

$$Y_i = \boldsymbol{c}_i^T \boldsymbol{X} = \boldsymbol{X}^T \boldsymbol{c}_i$$

Where the coefficients of $\boldsymbol{c}_i$ are to be learned by least-squares minimisation over a large number of $(\boldsymbol{X}, \boldsymbol{Y})$ pairs. This gives an equation of the form $\boldsymbol{Ax} = \boldsymbol{b}$ where $\boldsymbol{A}$ is a matrix of stacked $\boldsymbol{X}^T$ vectors and $\boldsymbol{b}$ is the corresponding Y_i values. Each such equation can then be optimised in a least-squares sense by standard algorithms and the resulting $\boldsymbol{c}_i$ vectors stacked in into the matrix $\boldsymbol{C}$ to give $\boldsymbol{Y} = \boldsymbol{CX}$.

Convergence

A good number N of random $\boldsymbol{X}$ samples to generate is $N = 500$ (fig. 7). The magnitudes of elements of a matrix generated $N = 5000$ is shown in fig. 3e, and its elements broadly line up with expectations from fig. 3d. The nonzero coefficients of the cart position can be attributed to the fact the matrix has not fully converged.

Single-step effectiveness of model

A model was trained with $N = 5000$ datapoints for this section [A3].

Expected vs target next step

To visualise how well the model matches the real system evolution over a single step, predictions can be plotted on the y-axis against the actual evolutions on the x-axis in pairs. The closer the points to the line y=x, the better the prediction. If an initial state is then scanned over, this forms a line which should ideally lie entirely near y=x. Such plots are shown in fig 8. They illustrate several features of the model:

- Predictions of θ and x are in general very good, while predictions of the velocities are much poorer.

- The model correctly demonstrates independence of θ and x on x and $\dot{x}$ (hence their lines are points).
- The velocities are slightly more sensitive to θ and x than the positions, but are still fairly independent.
- θ and $\dot{\theta}$ correctly have a large effect on state variables, but their predictions are fairly poor.

Single-variable scans

The fit is plotted for single variable scans against the target in fig. 4b, showing which relationships are modelled well and which aren't. Some subplots appear linear, but change over the initial state, so the model does not match them well.

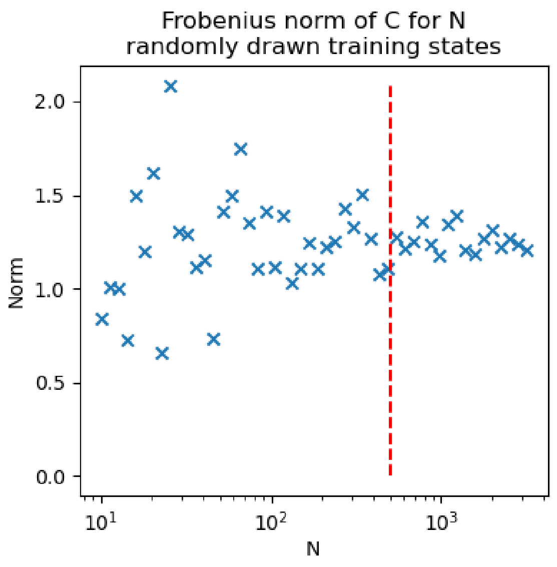


Fig 7. Frobenius norm of $\boldsymbol{C}$ matrix used as a metric for convergence, plotted for collections of $(\boldsymbol{X}, \boldsymbol{Y})$ pairs of varying sizes (with randomly drawn $\boldsymbol{X}$ within the variable ranges). $N = 500$ represents a good tradeoff of optimisation speed versus accuracy. The model is expected to perform fairly poorly, so a little noise is not much of a problem.

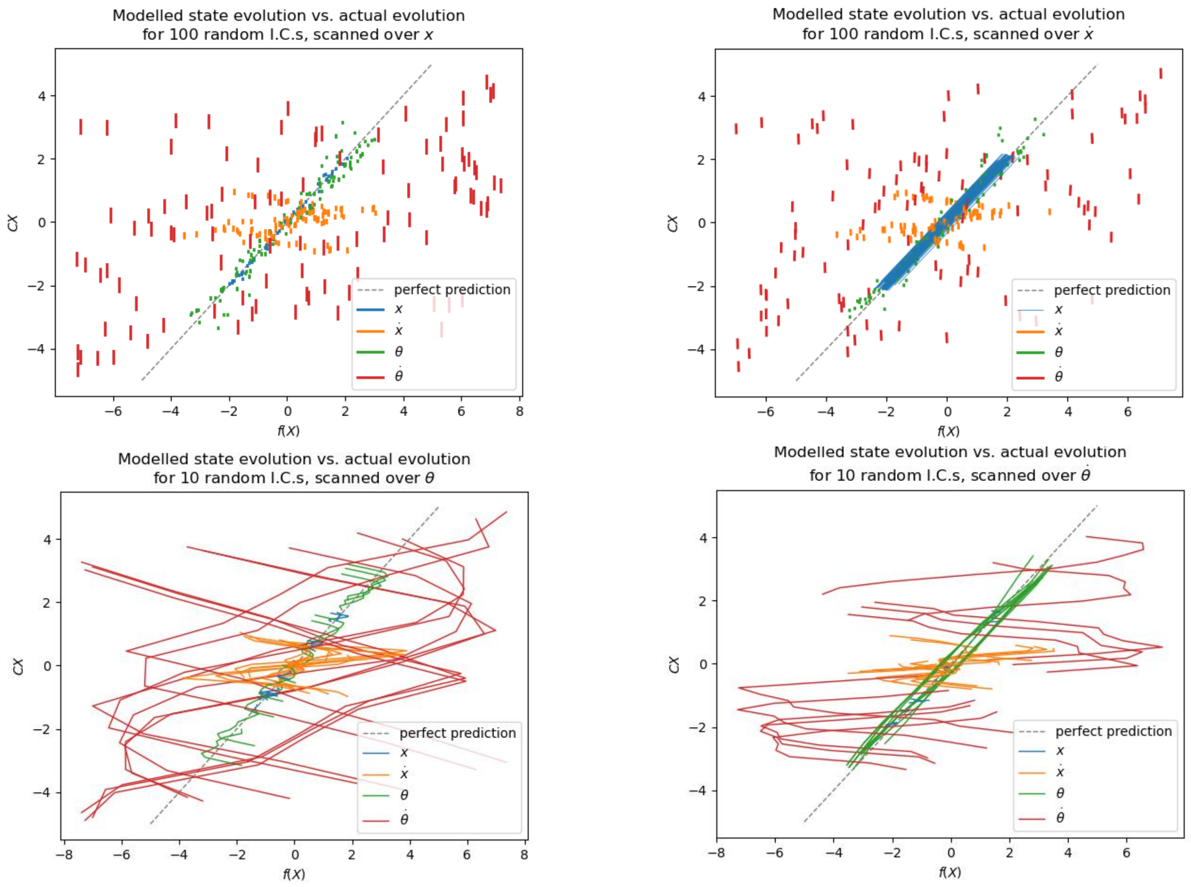


Fig 8. Comparison of state evolutions, initialised with random states, scanned over state variables.

Trajectories of model

In general, fitted models perform poorly. The state variables diverge from real trajectories within a few timesteps and sometimes immediately. The state variables (excluding the angle) eventually diverge to infinity, driven by the cart velocity, rather than decaying to zero, although this often takes a long time.

There are some situations when the model performs better. Some matrices, including the previously fitted one with $N = 5000$, reproduce simple oscillations qualitatively, even if they rapidly become out of phase and often have different frequencies. One particular pathology, which manifests even more strongly if the time step is decreased so that more evolutions are required per cycle (despite the simpler modelling task for a single evolution), is that the model unpredictably alternates between the bound and unbound states, ending up eventually in the unbound state (and then diverging), in contradiction to reality.

Conclusions

A linear model is insufficient to explain the evolution of the cartpole system, due to periodicity-induced nonlinearities in the angular variables which drive the system. A nonlinear model is required to fit the evolution function better.

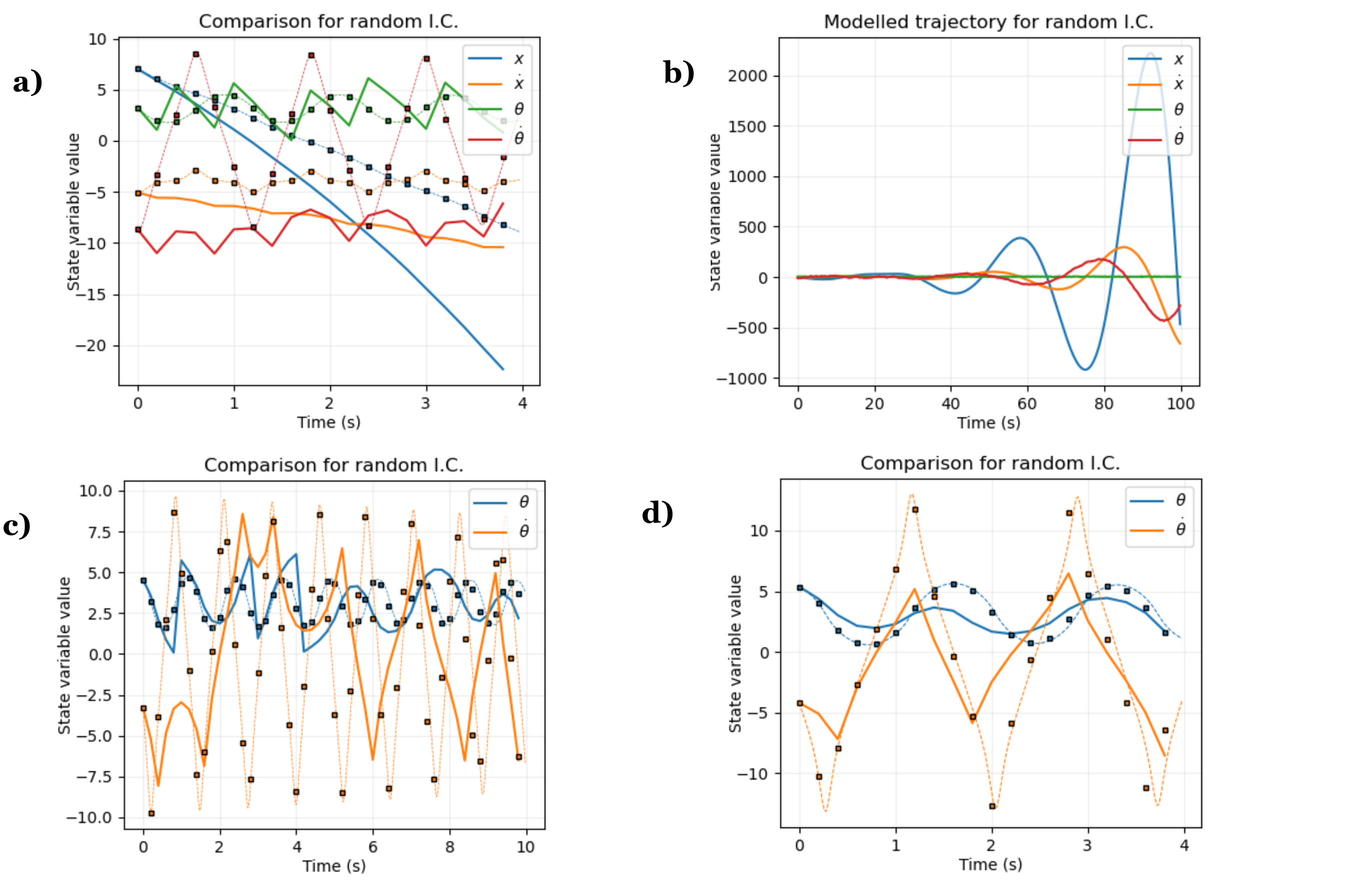


Fig 9. Trajectories predicted by the linear model plotted against true trajectories. **a)** a typical trajectory, **b)** eventual divergence, **c)** alternation between bound and unbound, **d)** good agreement for low energy.

Nonlinear Fit

Outline

To fit a good regression model to the update function, nonlinear kernel functions are required. For simplicity, diagonal-covariance Gaussians were chosen. Using a subset of training datapoints as kernel centres makes this correspond to a sparse Gaussian process. The solution can be regularised with hyperparameter λ to aid numerical stability.

Since Gaussians are better for fitting mean-zero fluctuations than linear trends, it's best to fit the nonlinear model to the residuals of the linear model. (Note: This requires enforcing zero θ-dependence in the $\boldsymbol{C}$ matrix to prevent periodicity artefacts.)

N and M, the number of training data points and the number of basis function centres, can be separately varied to give a trade-off of model complexity vs. fit, and the six hyperparameters λ, $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5]$ can be optimised for given data.

Hyperparameter Optimisation

Choosing initial values

First $N = 2048$, $M = 512$ data points were obtained with a Sobol sequence over the fitting region (same bounds as before).

From the plots of fig. 4, rough length scale variations can be selected as $\boldsymbol{\sigma} = [10^3, 10^3, 1,3,20]$ where the first two are already well fit by the linear model so should vary very slowly, if at all. λ was selected as

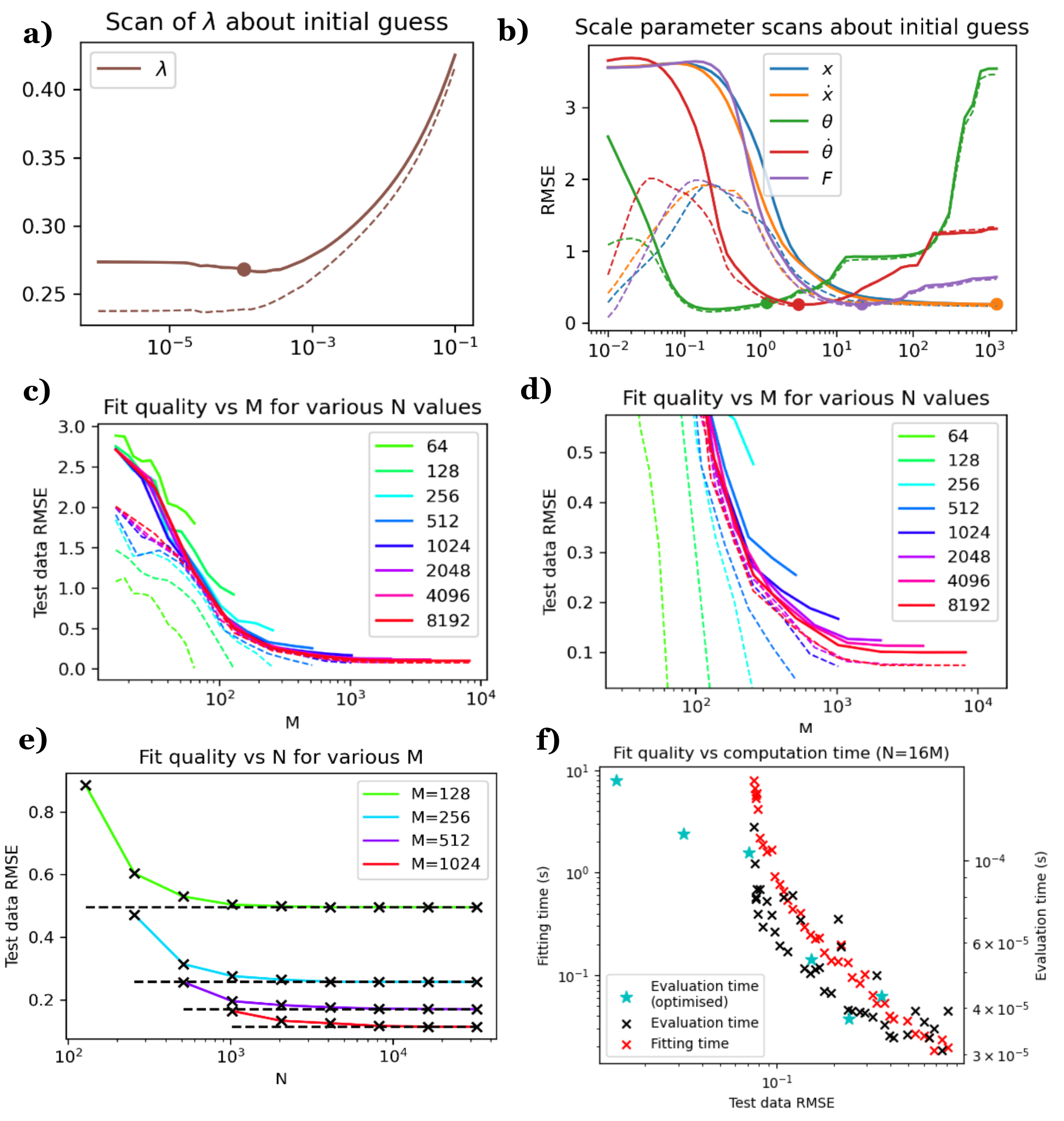


Fig 10. a), b): Hyperparameter scans about initial guess ($N = 2048, M = 512$). **c):** Lines with fixed training N in powers of two, varying M in powers of two up to N. **d):** as c) but zoomed in. **Dotted lines (a-d)**: test RMSE; **solid lines (a-d)**: training RMSE. **e):** Lines with fixed M, varying N in powers of two until asymptote. **f):** Scatter plot of fitting and evaluation times against test RMSE, up to M=2048. Re-optimising parameters for each M value helps break the asymptote, resulting in the trend line of blue stars. *In all cases,* $N_{test} = 8192$.

10^{-4} - it will worsen the fit when it is large by over-smoothing and smaller should be better (for the optimal solution) as long as the inversion is robust. Fit can be evaluated by generating N_{test} (normally 2^{12}) new random data points (not from the Sobol sequence to prevent systematic biases), and calculating root-mean-squared error. Scans from these initial guesses are shown in Fig. 10 (a, b).

Numerical optimisation

Starting with the initial guess, the Nelder-Mead algorithm was employed to (ideally) search for a good minimum. The results were:

$$\boldsymbol{\sigma} = [\, 408.8, 541.4, 0.267, 3.997, 19.76 \,]$$

$$\lambda = 10^{-2.878}$$

The fact that there is an optimal λ setting indicates that some smoothing is slightly helping the fit – with enough data, smaller should always be better. These hyperparameters should be good enough to assess fit convergence vs amount of data.

Fit vs amount of training data

Error should decrease with amount of data, and this is verified in fig. 10. The convergence of the model as M and N increase is shown in fig. 10 (c, d). For fixed M, increasing N leads to a limiting error at N≈16M.

Fitting time and model evaluation time must be considered against fit accuracy. Since N only comes into training & model evaluation, and because its computational burden in this range is small compared to that of the $M \times M$ matrix inversion, matrix N can always be set to 16M (this will be assumed from here). M should then be chosen based on a tradeoff between fit quality and fitting/evaluation time, plotted in fig. 10 (e). M=2^9 is good for comparison tasks where the model must be repeatedly fit (~ms to fit); M=2^{12} is better when a very good fit is required (~ms to evaluate). Re-optimising parameters for each M value, this time keeping the first two elements fixed at 10^6, improved fits further. This is because optimal scales are smaller when M in insufficient to recreate fine detail, so it is better to smooth over them. M in powers of 2 from 2^5 to 2^{13}, parameters were optimised and fits were created and stored for future use. These single-step fits are demonstrated in figs. 11 and 12.

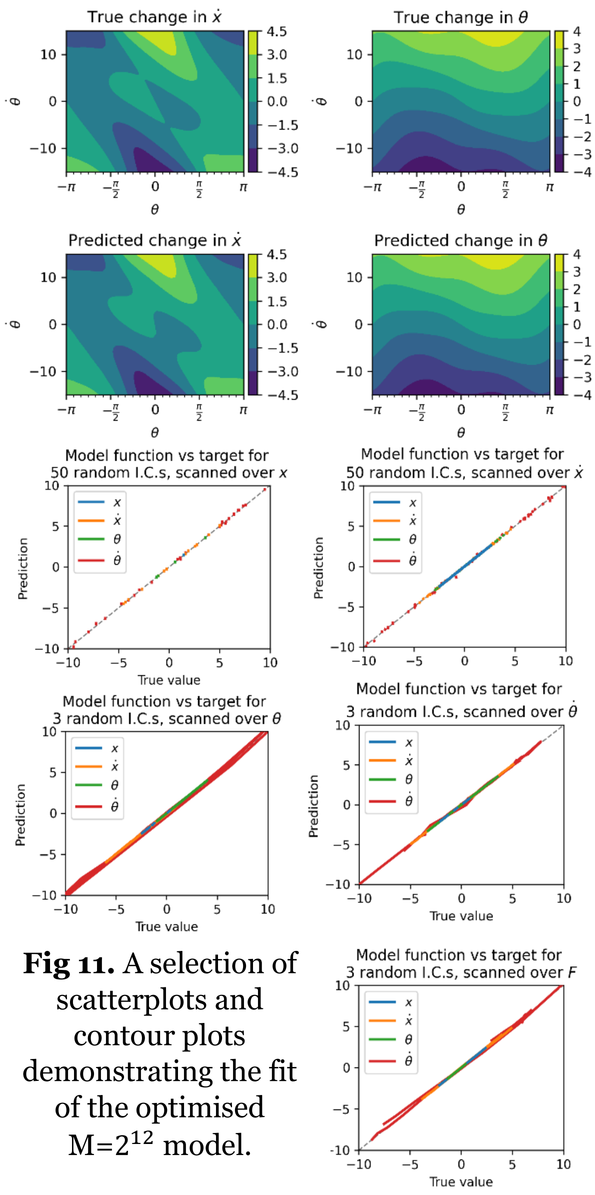


Fig 11. A selection of scatterplots and contour plots demonstrating the fit of the optimised M=2^{12} model.

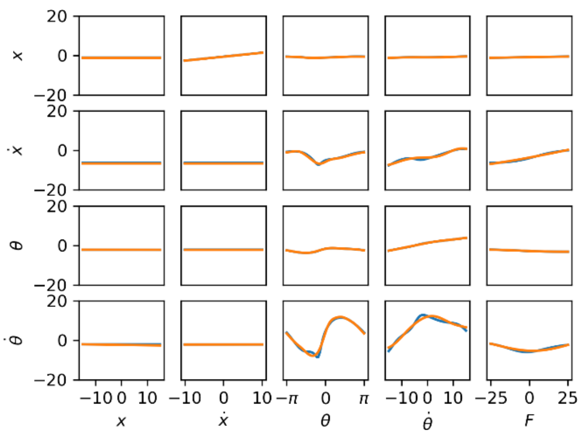


Fig 12. Single-variable scans demonstrating the fit of the optimised $M = 2^7$ model. Note how the smallest-scale variations as θ and $\dot{\theta}$ change are smoothed over. The $M = 2^{12}$ is not visually distinguishable from the original function in equivalent plots, so is not shown.

(Note: 2^{13} used parameters optimised for 2^{12} because optimisation would have taken days. The trend in optimal scales could have been extrapolated for this instead.)

Assessing rollout agreement

The nonlinear functions modelled rollouts extremely well in general – impressively so, considering the relative simplicity of the modelling process. Figure 13 shows some illustrative trajectories and typical agreements & disagreements.

Mismatch criterion

To quantitatively define agreement, a mathematical statement on at which point a rollout diverges was required. Several were experimented with, and the most reasonable was chosen as follows:

- Look backward for 10 iterations. Find the oscillation ranges of $\dot{x}$ and $\dot{\theta}$ over that window (for each of the first 10 states, the first 10-step window was used).
- Divergence occurs when the error in either exceeds a fraction f of its range, or when the error in x exceeds $15f$. (15 being half the x modelling range).

The fraction f was set at default to 0.2 (20%). θ was ignored because the dynamics of θ and $\dot{\theta}$ are so strongly coupled. In practice, the x criterion was never violated – errors in the other variables preceded this. The criterion is demonstrated in figs. 13 and 14.

Model performance

To evaluate a model's overall fit, iterations until mismatch were calculated over random ICs (capping energy to remain inside the modelling region). Histograms are shown in Fig. 14a, b. These could be combined into an overall score in a number of ways:

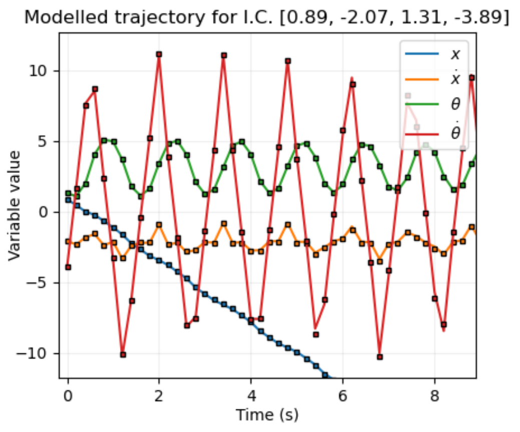
1. Mean number of matching iterations
2. Mean number of matching oscillations
3. Modal number of matching iterations
4. Minimum number of iterations for which 90% of rollouts still match

(1) is problematic because some trajectories match for a long time. Simulating until divergence would be computationally costly, and terminating at a maximum iteration will bias the mean down for these cases. (2) has the same issue. (3) is avoids the mean bias caused by termination, which is attractive.

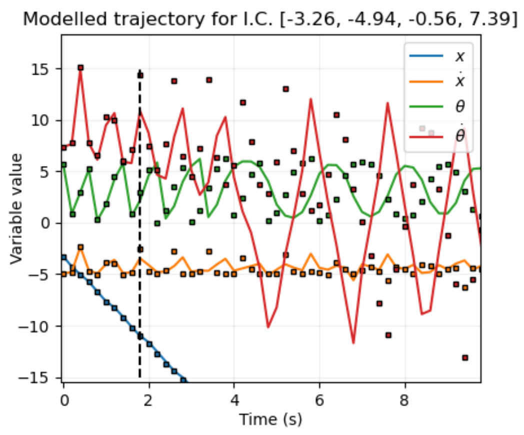
However, (4) is the most relevant measure to the task of control. Ideally, the system will be stabilised within a few cycles, and a good and *reliable* match is required in this time period – it doesn't matter if the average matching time is 1000 iterations if two thirds of cases diverge after 3. This score provides the added benefits of requiring much less computation (since there is no need to simulate for long time window to see what happens to well-agreeing trajectories), and being insensitive to the precise value of f (since most trajectories diverging early diverge by a lot – fig. 13b, 14a). The threshold of 90% was chosen to give a reasonable output spread for model comparison.

From figs. 14c and 10f, $M = 2^{11}$ was chosen as a reasonable tradeoff of evaluation time (~0.1ms) and score (~6.5 iterations until 10% diverge) for control. For reference, the actual system dynamics function as provided took ~.7ms to evaluate, and a C-accelerated version that was written to aid computation took ~0.01ms.

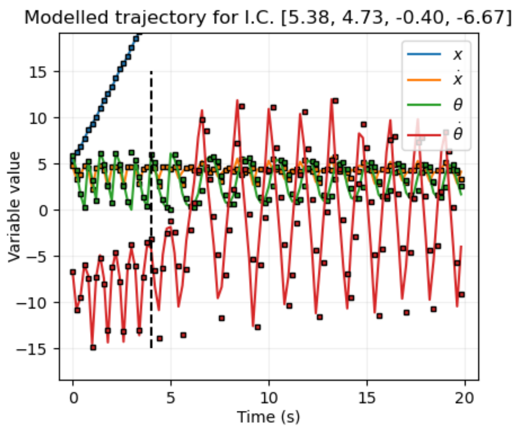
a)



b)



c)



d)

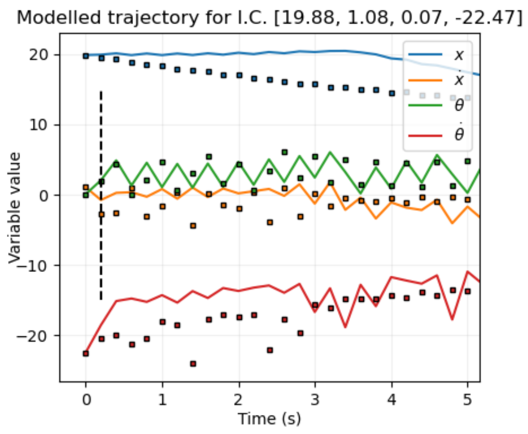
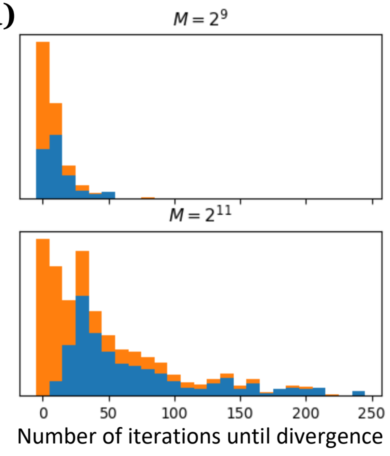
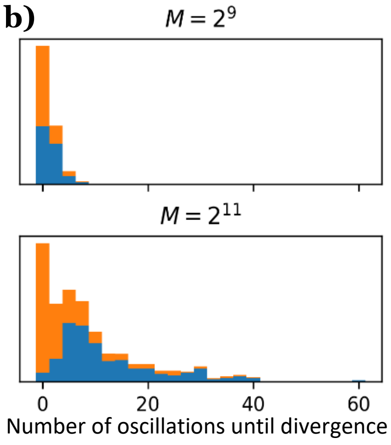


Fig 13. Rollouts with illustrative initial conditions, using a model with M=2048. **a)** Bound states were generally modelled extremely well, matching for hundreds of timesteps (the precise mismatch point is then irrelevant for the purposes of control). **b)** Early divergence was most common around decay from unbound to bound states. **c)** The mismatch threshold was set relatively strictly, so that some trajectories remaining qualitatively similar were considered to diverge. **d)** Energies were generally constrained to ensure states remained in the modelled region. When outside, the model performed predictably poorly. *Black dashed line*: Step at which trajectories were considered to divergence; mismatch threshold set at 20%.

a)



b)



c)

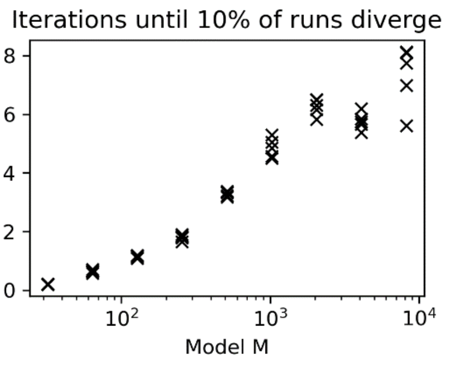


Fig 14. a) Iterations until divergence, $f = 0.2$, for models with $M = 2^9$ and 2^{11} kernel functions. *Orange:* unbound ICs, exponential-like distribution. *Blue:* bound ICs, peaked distribution. These different shapes are due to the fact that the transition from unbound to bound is the most error-prone. **b)** Same plots, but quantified in number of oscillations *(as required)*. Unbound cycles were considered half-oscillations. **c)** Model matching score for optimised models with M increasing in powers of 2, each run 5 times for 500 trials. Note that one oscillation is approximately 8 iterations.

Linear Control

Outline

Problem formulation

The simplest way to control the cart is with a force that is a linear function of the state variables:

$$F = \boldsymbol{p} \cdot \boldsymbol{x} = p_1 x + p_2 \dot{x} + p_3 \theta + p_4 \dot{\theta}$$

Linear controllers about equilibrium points are well-studied and can be obtained formally through the equations of motion. The linear policy will instead be obtained instead in a data-driven way, in preparation for the harder problem of nonlinear control.

To find an effective policy, a score must be given to each set of policy parameters based on how well it stabilises the cart. This is defined as the loss function

$$L = \sum_{i=1}^{N_{it}} l(\boldsymbol{X}_i)\, ;\ \text{with}\ l(\boldsymbol{X}_i) = 1 - e^{-\frac{1}{2}\sum_{j=1}^{4} X_{ij}^2/\sigma_j^2}$$

With scale parameters σ_j for each state vector component, a window of N_{it} iterations, and target state $\boldsymbol{X} = \boldsymbol{0}$. Although a linear controller is not expected to stabilise initial states far from the equilibrium, there is no harm in also introducing periodicity to the loss function in the same way as for the nonlinear model (by using a sine function for the θ term).

The policy should ideally be:

1. **Robust:** stabilise over a wide range of initial states.
2. **Fast:** stabilise reasonably quickly.
3. **Bounded:** Avoid visiting extreme states, i.e. states outside the modelling range.

The loss of a *policy* should then be defined as L evaluated over a relatively small number of iterations N_{it} (to ensure goal 2), evaluated as an average value over a number of random perturbed initial states (to ensure goal 1). If the trajectory diverges outside of the modelling range, the loss should return its maximum value of 1 (satisfying goal 3). Since restricting the time allowed for convergence reduces robustness, and considering the iterations until divergence in Fig. 14c, N_{it} was set to 20.

Loss discount

This formulation of loss function punishes certain types of policy – for example, if the pole starts with $\theta = .001$ and $\theta = 10$, it may be desirable to let the pole swing around a full revolution before stabilising, but this will pass through a high-loss region. One way of allowing this type of policy is to multiply cumulative loss by a discount factor γ every timestep, so that early contributions are reduced.

However, this will not be used for the linear controller, on the assumption that it is not complex enough to create this kind of policy while still being good in other regions of parameter space.

Note that the logarithm of the loss was used for optimisation for computational reasons.

Strategy

The global minimum basin in $\boldsymbol{p}$-space giving good policies is likely to be very small, possibly hard to find, and one of multiple local minima, even when loss is evaluated using the true system dynamics. When using the modelled dynamics, whose deviations from the true evolution are nontrivial, the loss landscape will be more complex and the global minimum may not coincide with the previous. These problems will reduce in severity as the fit converges (with high numbers of basis functions).

In a real-world situation, the goal is then to find a fit good enough that it can be used to obtain a robust linear policy, while having few enough basis functions that evaluating the function during optimisation is computationally feasible. Here, this will be explored by:

1. Optimising using the true dynamics to find ideal policies
2. Optimising using modelled dynamics for various model sizes
3. Comparing quality and robustness of policies obtained using these models.

Different linear policies be effective starting from different regions of parameter space. Policies which start only at small deviations from the target state will be explored.

True dynamics optimisation

Getting an initial good policy

This is still a hard, multimodal problem with a small region of good policies and large region of terrible policies. To find the former,

1. Loss scales should be as large as possible (to encourage a more gradually-sloping loss landscape, assisting with locating minima) while being small enough to punish bad policies. Initially, scales were chosen to have $L \approx .99$ when each component is at its range limit and the others are zero ($\boldsymbol{\sigma} = [5, 3.3, 0.3, 5]$). However, policies derived from this tended to have x and $\dot{x}$ converge much more slowly than θ and $\dot{\theta}$. To punish this, σ_1 was set to 1 instead. (*Note: the scales could have been explored in more detail with plots and optimisation but were not due to time constraints.)*
2. The problem was split into easier problems, the solutions of which were nested supersets of the overall solution: the penalties for $\dot{x}$ and x were originally neglected, before being reintroduced (in that order).
3. IC ranges were initially set small, and manipulated as a fraction of the overall parameter ranges f_{IC}.

With $f_{IC} = 2^{-10}$, with $x, \dot{x}$ losses neglected, there was a clear wide global minimum basin in the $\theta - \dot{\theta}$ plane (fig. 17ai). As f_{IC} increased its 'walls' rapidly became steeper.

When the $\dot{x}$ loss was introduced, there was still a clear dominant basin, as verified by scanning through one of the angle variables across the first basin while taking 2d contour plots of the other two (fig. 17aii, iii). A line was fitted through the deepest valley in the $\dot{\theta} - \dot{x}$ plane, x loss was introduced, and θ was

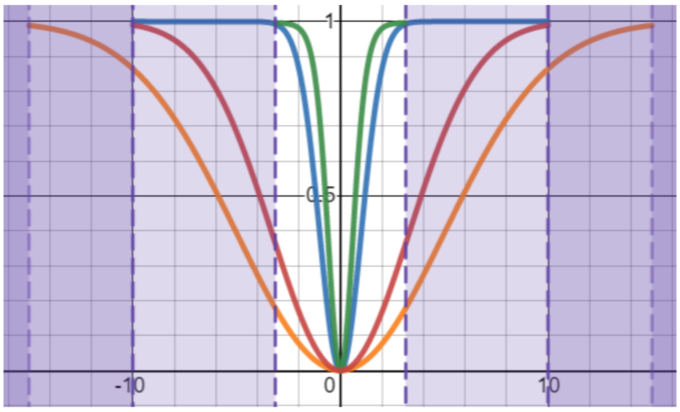


Fig 16. Loss scales demonstrated. Blue: x scale, orange: x scale, green: θ scale, red: $\dot{\theta}$ scale.

scanned through while making contour plots of the combined velocity variable and x, revealing that a clear dominant basin once again remained (fig. 17aiv). The overall volume of parameter space which contained minimally useful policies was approximately

$$\boldsymbol{p} = [-2.5, -2.5, 5, 0.5] \text{ to } [3.5, 2.5, 25, 3\,]$$

Testing robustness

A good policy should stabilise a wide range of ICs. To check the robustness of a policy, the following algorithm was used:

1. For each state variable i, set the random IC range to zero for every other variable, then gradually increase f_{IC}^i between 0 and 1.
2. For each f_{IC}^i value, run 100 random IC tests, and calculate the *success rate*, evaluated by whether the run spends at least 10 iterations with mean state magnitudes less than 10^{-8} , allowing a total of 50 iterations for this to happen. *(Note: this cannot be itself used as a loss function, as there are a discrete number of runs, and it would be computationally infeasible to have enough that numerical optimisation is possible).*
3. Record the first f_{IC}^i value at which the success rate is not 100%, f_{crit}^i.
4. Once this is done for each state variable, carry out the same procedure, but using f_{IC}', a fractional value that multiplies the vector of parameter thresholds found in step 3.

5. The percentage of modelled state space which is stabilised is then approximately $f_{tot} = f'_{crit} \cdot \prod_i f^i_{crit}$.

Note on success rate

A 100% success rate was required – this is strict, but a policy that only succeeds, say, 70% of the time is not very useful in practice, and there are theoretical guarantees on the linear model's performance nearby the optimum.

Finding robust policies

Two strategies were used to search for robust policies.

Firstly, the manually-located minimum gave a good starting point for optimisation within a simple basin, using the Nelder-Mead method. Noise due to random ICs hindered this by creating many tiny local minima, so a set random seed was used for each evaluation point in a given optimisation run, meaning that the same set of (50x) ICs were used over the course of that run (leading to a small potential bias at convergence).

Once the method converged to a minimum, slices were made about that point in all six planes to get a rough picture of the loss landscape around it (fig 17c). The robustness of this policy was probed via the above strategy.

It is possible that a model optimised on a wider IC range might be more robust to ICs outside of this training range. To test this, f_{IC} was increased by factors of $\sqrt{2}$, the loss function re-optimised (starting from the

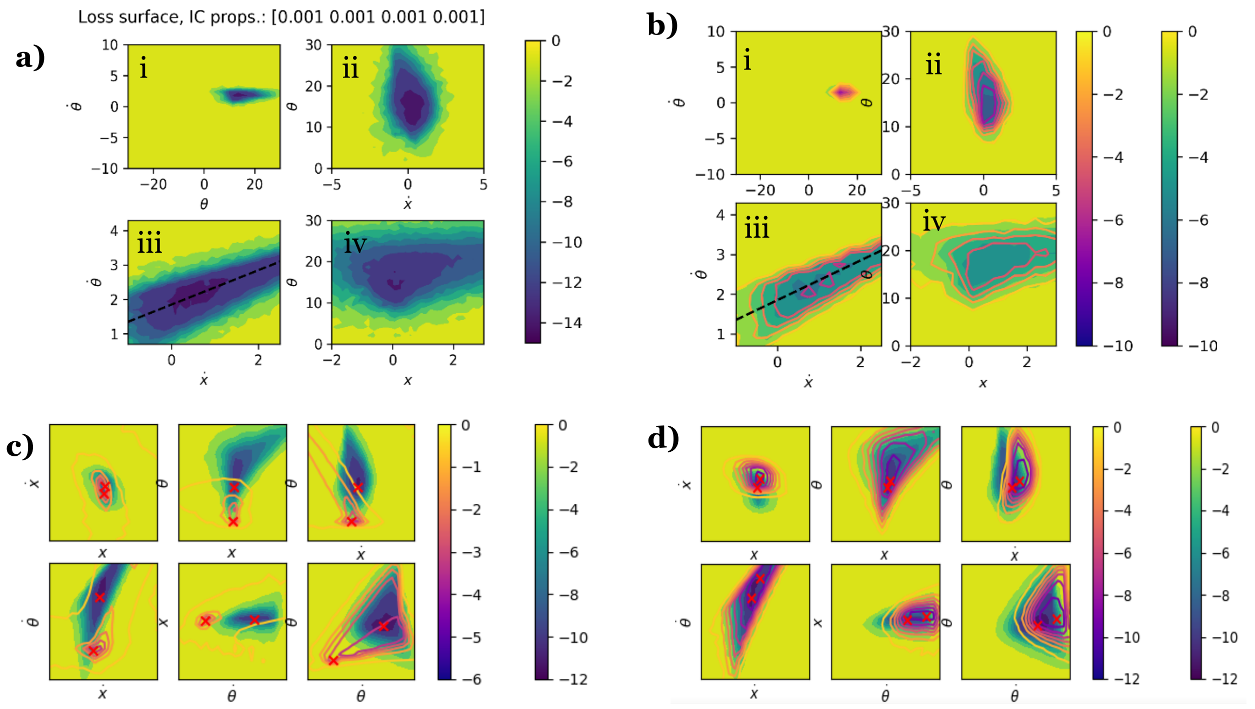


Fig 17. Contour plots of loss surface, refining from simplified problem to full problem. **a)** i) Only angular variable losses considered. ii) One of the fixed-$\dot{\theta}$ scans located around the minimum, with the $\dot{x}$ penalty introduced. iii) an equivalent plot with fixed θ. Dotted line is the enforced linear dependence held fixed in iv), which introduces the x penalty. *Note that noise is present (particularly in this subplot; which used 20 random runs, while the others have 50) and hinders optimisation due to different ICs.* **b)** M=2^7 model, superimposed with corresponding plots where $f_{IC} = 0.01$. *($f_{IC} = 0.01$ for **c** & **d** too.)* **c)** Planes scanned around the minimum found by optimisation. Landscape was featureless outside of this region. Superimposed lines show contours of $M = 2^7$ over the same regions. Optima for both are marked. **d)** As with **c**, but plotted for $M = 2^{13}$. The match is still not perfect but the overlap is much greater.

previous minimum), and these plots and robustness tests repeated, until $f_{IC} = 1$. (fig 18a).

Secondly, optimisation was attempted at the same set of f_{IC} values, but with random initial $\boldsymbol{p}$ vectors where each element $\sim U(0,20)$. 25 attempts were made for each f_{IC}, and the lowest loss function result taken. Results are in Appendix A4.

Conclusions

The most robust policies had the optimisation f_{IC} set to $\sim[2^{-4.5}, 2^{-2.5}]$ approximately. However, these policies were very hard to find: The random initialisation optimisation attempts found policies that matched the performances in fig 18a with a probability of 1/6 for small f_{IC}, and runs which didn't find these optima gave policies that performed Robustness hermore, this probability reduced as f_{IC} increased to the point where that no successful policies were located for any $f_{IC} \geq 2^{-7}$.

One way to subvert this problem was to set f_{IC} 2^{-10} or smaller, perform optimisation with random ICs to find the minimum, and use this result as a starting point for optimisation on $f_{IC} = 2^{-4}$. Alternatively, using the simplified, encompassing problems could be used to graphically find a starting point.

Policies found for smaller f_{IC} did not perform significantly worse in terms of robustness, so random initialisations not converging is not an big issue. However, for modelled behaviour or a noisy system, random initialisations can be expected to perform significantly worse, so the graphical approach is favoured if possible.

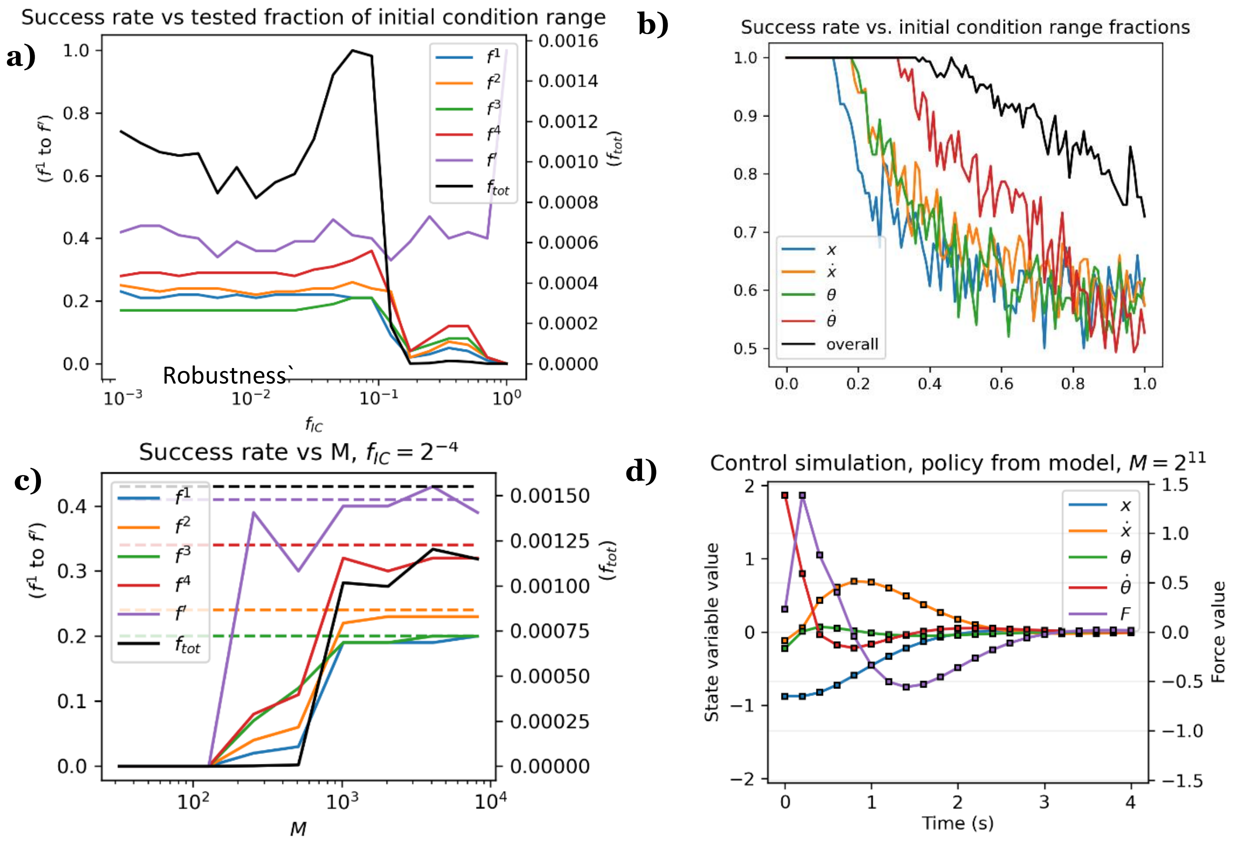


Figure 18. Robustness of the optimized policies, and typical plotted rollouts.
a) Robustness evaluated for policies optimised at different f_{IC}'s. **b)** Demonstration of robustness calculation by finding the threshold of a 100% success rate in each variable, then all variables combined. **c)** Robustness of policies optimised from modelled system dynamics at different complexities M of the model. *Dotted line:* Best robustness found in **a**. **d)** A typical control simulation showing a policy derived from the $M = 2^{11}$ model converging on the goal state.

Model dynamics optimisation

Landscape visualisation

For the modelled dynamics, the loss landscape is likely more complicated and multimodal than for the true dynamics. However, plotting graphs for the simplified problem for M as low as 2^7 (in the same manner as before) reveals a very similar structure (fig. 17b). This means an optimisation start point is easy to find even if the landscape becomes more complex.

Optimisation was performed for $\log_2 M \in [6,7,\dots,13]$ with $f_{IC} = 2^{-12}$, using the previously obtained rough manual start point. Contour plots in the six possible planes of two variables were made about this point, and superimposed on those of the original model. (fig 17c,d) Once again, although the basins were more complicated, they broadly aligned in behaviour, even though they didn't entirely coincide.

Performance vs M

For each M value, optimisation was repeated with $f_{IC} = 2^{-4}$, and the robustness of the optimisation results probed as before. (fig 18c)

For $M \leq 2^7$, controllers were unstable. All the other controllers were able to stabilise the cartpole from some initial states, and performance approached the ideal case from $M = 2^{10}$, showing that model-predictive control is extremely feasible for this system. (fig 18d). By inspecting cases where the models failed, it was clear that either oscillations were not killed fast enough, or the system went unstable, rather than (for instance) nonzero final cart position.

Conclusions

Obtaining a useful linear policy entirely from a nonlinear is easily within the realms of computational feasibility, as long as sufficient data is available to train the model.

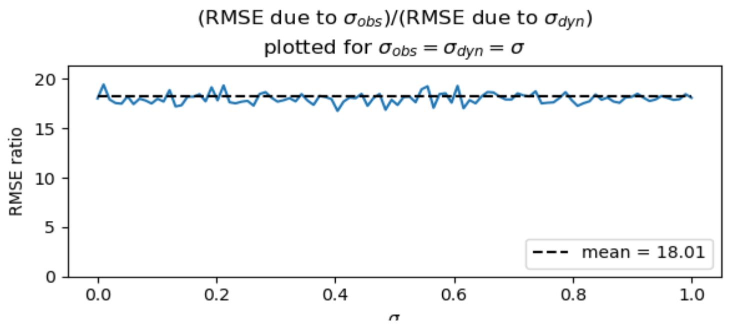


Fig 19. Calibration of dynamical noise to observation noise.

Effects of noise

Outline

In a real-world scenario where model-predictive control is to be used to stabilise the cartpole, noise will be present, which will degrade both the model and the policy. It is important to understand the effects of this noise to be confident in the reliability of a controller, with the goal in mind of learning how much noise can be tolerated for a robust control system, and how

Two principal types of noise will be investigated:

1. Noise in observations of the model *(for example, due to a camera's finite precision, or an inertial measurement unit's inherent measurement noise)*
2. Disturbance to the dynamics of the model itself *(for example, due to wind or vibrations applying random forces).*

For (1), this is often due to a large number of small random factors and can be well-modelled by an additive i.i.d. Gaussian term in observations. The standard deviation of noise will be controlled with the single term σ_{obs}, the fraction of half the modelled range for each parameter. There could also be an unknown bias, but due to time constraints this will not be explored. Observation noise was also added to F.

For (2), there will an additive term to the forces in the equations of motion, which will have a nontrivial power spectrum – e.g. a sum of random-phase sinusoids. However, this will be again modelled as i.i.d Gaussian white noise (as a first approximation) added at each Euler iteration. A Gaussian with the same variance σ_{dyn} is added to both the $\dot{\theta}$ and $\dot{x}$ update equations for simplicity. Once again, a constant term could be explored but won't be due to time constraints.

Dynamical noise calibration

The parameter σ_{obs} has a straightforward interpretation, but σ_{dyn} is more arbitrary. To give it some link to reality:

1. The standard deviations of the right-hand sides of the acceleration update equations were calculated over many random start states for a single step, and the scale of Gaussian noise added to each was set to this value multiplied by σ_{dyn}.
2. The RMSEs of σ_{obs} and σ_{dyn} were compared (fig. 20), resulting in the relation that effects of σ_{dyn} and σ_{obs} are comparable over a single step when $\sigma_{dyn} = 18\sigma_{obs} \approx 20\sigma_{obs}$.

Effects on single steps and rollouts

Figure 20a,b show scans of the target function with added observation and dynamics noise. The effects of σ_{obs} aren't surprising, but noteworthy for σ_{dyn} is the larger amount of noise on velocities than positions. This is due to the smoothing effect of integration on high-frequency noise. (Note that realistic noise may have its power concentrated at lower frequencies than white noise.)

Figure 20c,d shows the evolution of a representative start state under the two types of noise. Immediately clear is that σ_{dyn} causes much greater divergence than σ_{obs} in the long run, since the underlying dynamics are cumulatively affected.

Effects on linear model

As σ_{obs} increases, the fit unsurprisingly worsens – however, not by much (fig 21a). Increasing the number of training points N allows the model to perform almost as well as it did in the noiseless case – with $N = 2^{14}$ (very achievable) noise all the way up to 20% of the modelled ranges can be almost entirely rejected, while a modest $N = 2^{10}$ rejects noise well up to $\sigma_{obs} = 0.1$. This is due to the least-squares fitting process, which is ideal for filtering Gaussian noise. The finite

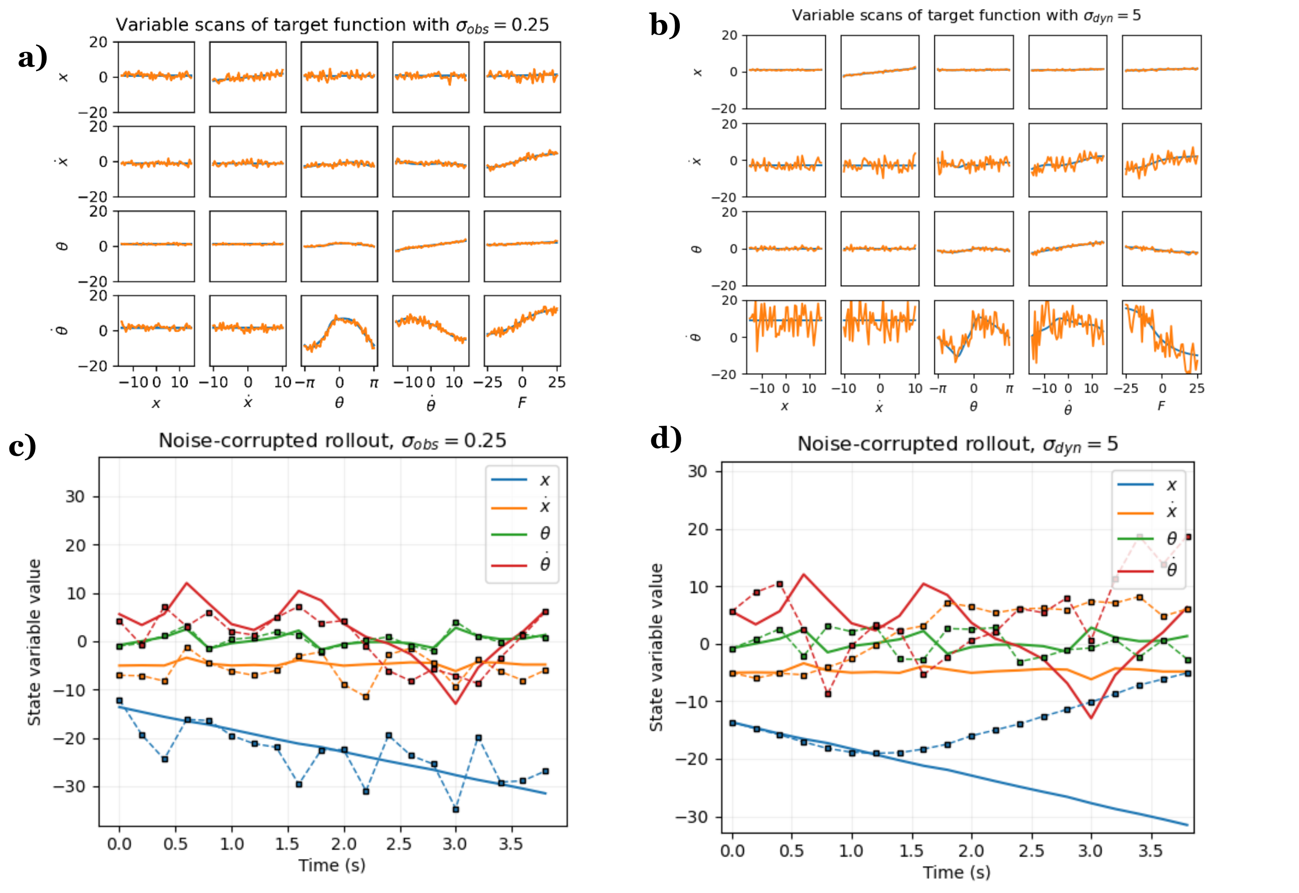


Fig 20. Visualisation of noise effects.
a) Moderate observation noise. Note that for training, there is noise both on the starting state and finishing state. There is only noise on the finishing state in this plot. **b)** Analogous noise level for dynamic noise. **c), d)** Noisy rollouts. Solid lines are noiseless system's trajectories from the same ICs, dashed/marker lines are noisy trajectories.

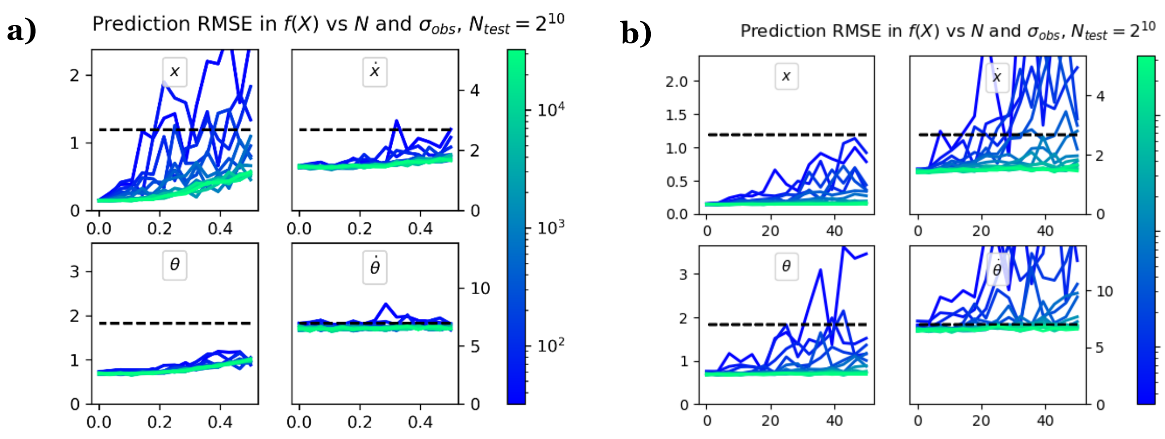


Fig 21. Effect of noise on linear model. a) observation noise, **b)** dynamics noise.

measurement range means that as the noise becomes comparable to this range, the distribution of values approaches uniform and all terms are modelled as the mean, i.e. zero, resulting in significant loss of fit. However, noise this high is unreasonable in practice and would make the system impossible to control.

Disturbance to the dynamics has a similar effect (fig 21b), without the issue of sampling range since the noise is within the model. $N = 2^{14}$ still gives practically the same model all the way up to $\sigma_{dyn} = 50$.

The effects of noise on modelled trajectories will not be discussed fully, since they are already terrible in the noiseless case – however, it is clear this performance can be recreated easily with sufficiently large N.

Effects on nonlinear model

Choice of M

Noise does not increase the complexity of the modelled function in theory, in terms of the number of basis functions required to approximate it. (In fact, if insufficient data is available to recover the model, noise decreases the complexity by smoothing).

Since $\mathrm{M} = 2^{10}$ was the minimum M able to create a linear policy whose robustness was approximately equal to the asymptotic value, $M = 2^9 - 2^{11}$ will be considered for analysis in this section (no higher for computational reasons).

The required value of N for convergence, however, will now be greater than before. Luckily, this larger data is only required for training, so is a one-off computation.

Effects on hyperparameter optimisation

Starting with the same initial guess as previously, scans were made of λ and the scale parameters under different noisy conditions and N values. Unlike previously, there was an optimal value of λ for intermediate N and noise, both for the noisy test data and for noiseless test data. This was because it could provide smoothing. However, as N increased, the λ curves took a similar form as the noiseless case, and the score with noiseless test data approached the noiseless case.

For both types of noise, scale parameters' scans' basins 'flattened' as noise increased, indicating that their precise values mattered less with increasing noise. (Fig. 23 a-e)

Because of this, the previously optimised set of parameters were used to analyse the convergence of the model with N.

Convergence with N

For $M = 2^9 - 2^{11}$, N was increased up to computationally feasible limits. This was done for

$$\sigma_{obs} \in [0.001, 0.002, 0.005, \dots, 0.2, .5]$$

$$\sigma_{dyn} \in [0.02, 0.05, 0.1, \dots, 5, 10]$$

To cover the full range of noise behaviours.

For $\sigma_{obs} \leq 0.01$, the original fit could be recovered with sufficient N. $\sigma_{dyn} \leq 0.2$ was the corresponding limiting value for dynamics noise (Fig. 22f,h). These relatively large tolerable noise levels are impressive, and are due to the nature of Gaussian processes and their ability to reject Gaussian noise.

For larger σ_{obs}, the original fit could never quite be recovered, and with $\sigma_{obs} \geq 0.1$, even the asymptotic value appeared to be significantly worse than the noiseless fit. The same was found for dynamics noise, but with the latter effect starting at a higher corresponding value of $\sigma_{dyn} \geq 5.0$, its better performance indicating this could be partially due to edge effects of the modelled range, similar to the linear model (Fig. 22g,i).

The value of N/M at which convergence occurred was picked out manually across a set of graphs and plotted against noise (Fig 22j,k). The value $N = 64M$ was chosen as a tradeoff with computational cost.

Testing optimal fits from noisy data

As discussed, hyperparameters optimised in the presence of noise should provide a slightly, but not significantly, better fit. Hyperparameters were optimised for $M = 2^{10}$, $N = 2^{16}$, and noise over the previous ranges. These fits were tested in their single-step fit to the noiseless system and ability to predict trajectories. (Fig. 23)

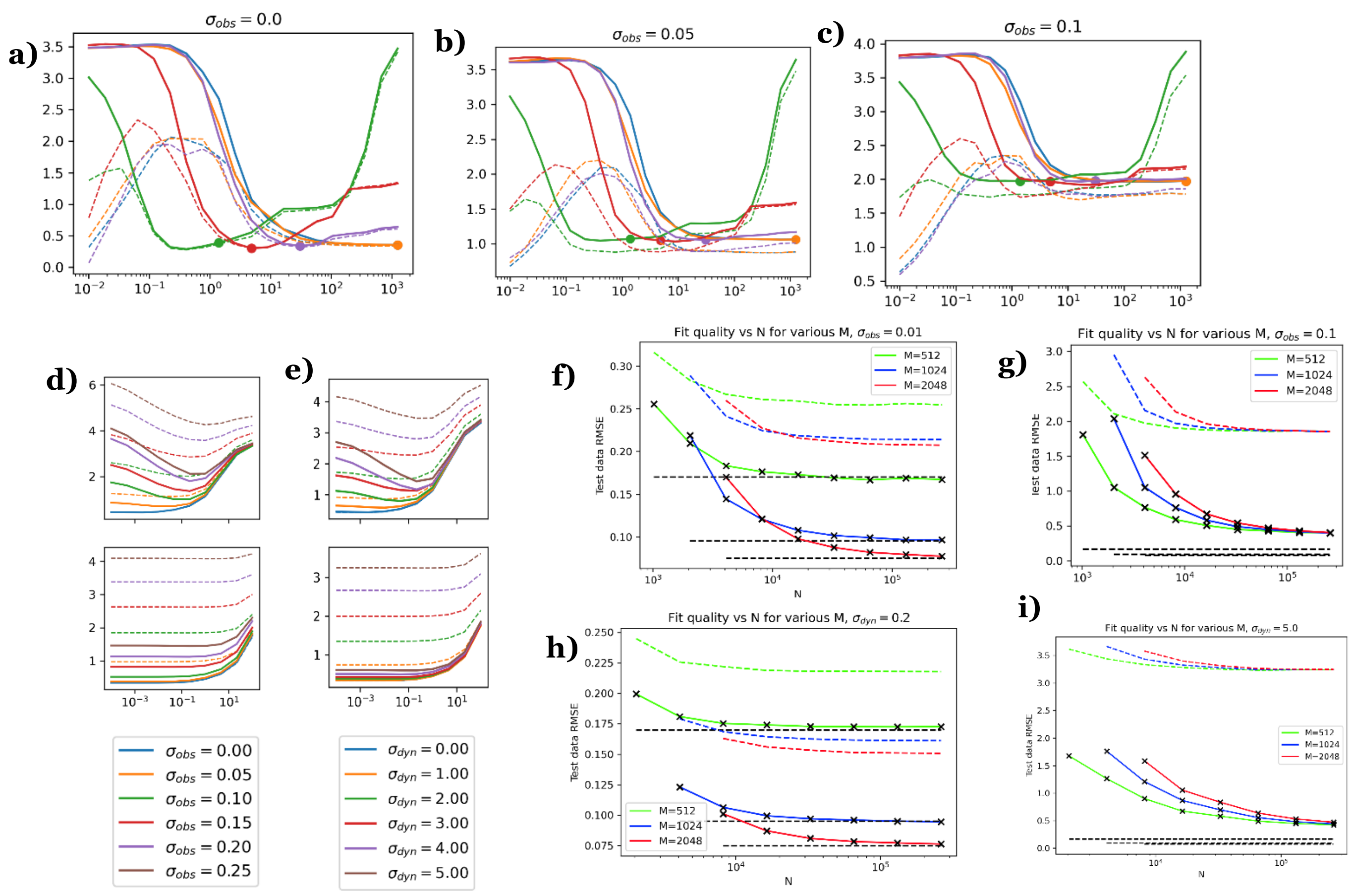
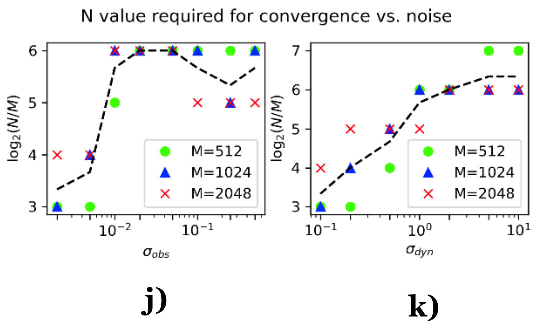


Fig 22. Effects of noise on nonlinear model



a-c) Parameter scans for different observation noise levels. Plots were analogous for dynamics noise. *Noisy test RMSE solid and training RMSE dashed*.

d,e) Minima appearing in λ scans for increasing noise levels in observation and dynamics noise, respectively. *Noisy test RMSE dashed and noiseless test RMSE solid.*

f-i) Convergence with N for selected noise values. *Noisy test RMSE dashed and noiseless test RMSE solid. Black dashed lines are optimal noiseless fits.*

J,k) Mean manually picked factor that N must be greater than M for, for observation and dynamics noise respectively. *Dashed line: mean across M =9-11*

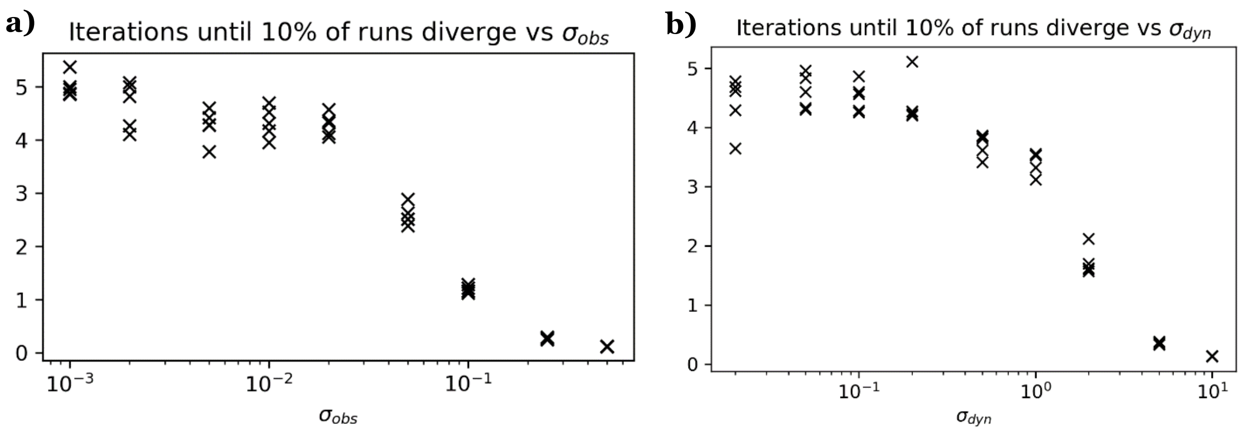


Fig 23. Effect of noise on nonlinear rollout predictions. a) observation noise, **b)** dynamics noise. The thresholds at which predictions degrade correspond to the thresholds discussed, shown in Fig. 22. Noiseless rollouts were tested against, because in the dynamics noise case trajectories would otherwise diverge regardless of how good the model is.

Effects on linear policy

Initially, the true system evolution will be used for analysis as before, to see how a noisy system affects the best policy created in the absence of noise.

Effects on previous best policy

For small amounts of noise, the previously obtained best policy still worked. At equilibrium, the system effectively became a linear filter from the input noise to the perturbations to the system (fig 24, column a). Noise gains in steady state, calculated as $g_i = \frac{\sqrt{Var(X_i)}}{Range(X_i)} \div \sigma_{obs}$, over many stable σ_{obs} values, were found to be:

$$\boldsymbol{g_i} = [\,2.2, 7.4, 4.1\,, 8.7] \pm [0.1, 0.4, 0.3, 0.3]$$

and the equivalent for σ_{dyn} (multiplied by 18 for comparison with σ_{obs}) was

$$\boldsymbol{g_i} = [\,1.8, 4.8, 2.5, 4.1] \pm [0.2, 0.3, 0.2, 0.2]$$

(the $\pm$ refers to standard deviation in values).

Therefore, convergence was defined as when $\frac{\sqrt{Var(X_i)}}{Range(X_i)} \div g_i$,over the last 25 observations and averaged across the four states, was less than $1.5\sigma_{dyn/obs}$.

Policy robustness (as computed before) is plotted against noise quantity for this existing controller in fig 24, columns b-c.

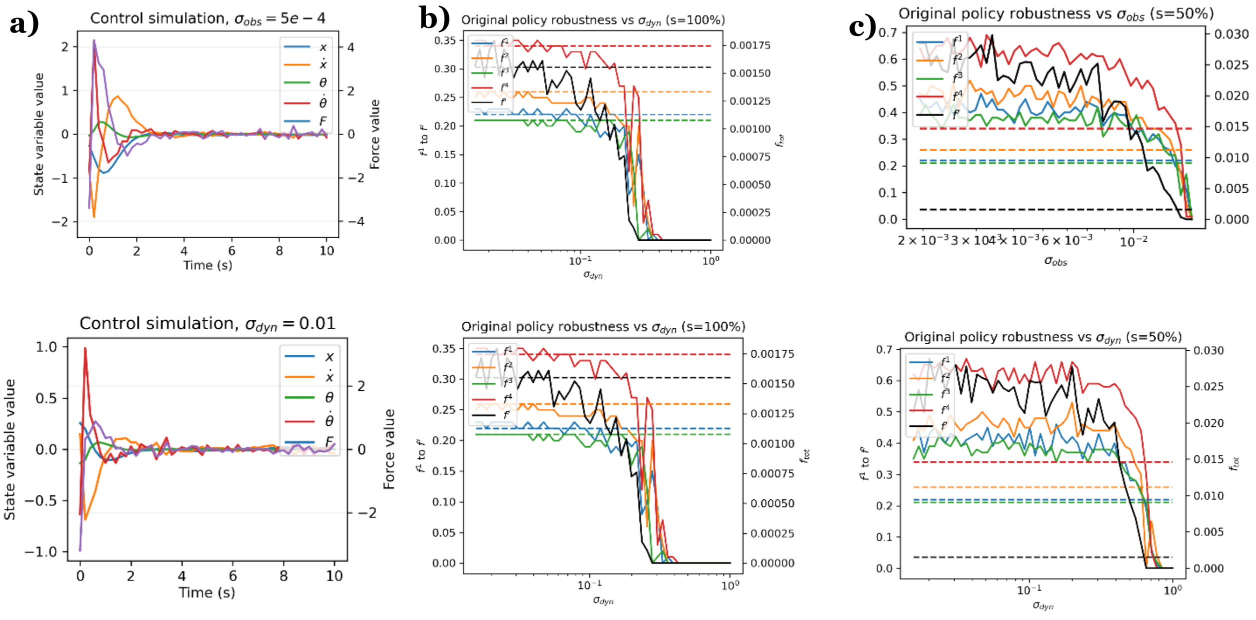


Fig 24. Noise effects on previous best linear policy.

Divergence thresholds are analogous for the two types of noise. Cutoff thresholds, at which the system is too noisy to control well, are sharp for a success rate of 100%, although they were slightly smoother for a 50% success rate (which is not very useful in practice).

In a real-world situation, the policy optimisation process would have to be done in the presence of noise. This could result in policies which better reject the noise, even if their robustness in the noiseless case is inferior.

Optimisation in the presence of noise

The contour plots used to locate the starting point for optimisation still worked for the maximum levels of noise at which policies were effective, so this is not an issue in the real world when it wouldn't be known in advance from the noiseless case (fig. 25).

Conclusions

The effects of the two noise types were very similar on modelling and policy learning, despite their different origins. For both, models and policies could be fit with reasonable computation time up to a fairly sharp threshold, beyond which reinforcement learning and model predictive control are

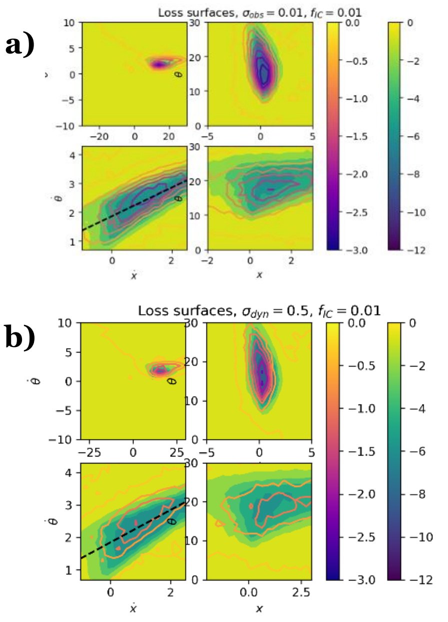


Fig 25. Initial guess with noise, using contour plot visualisation as before.

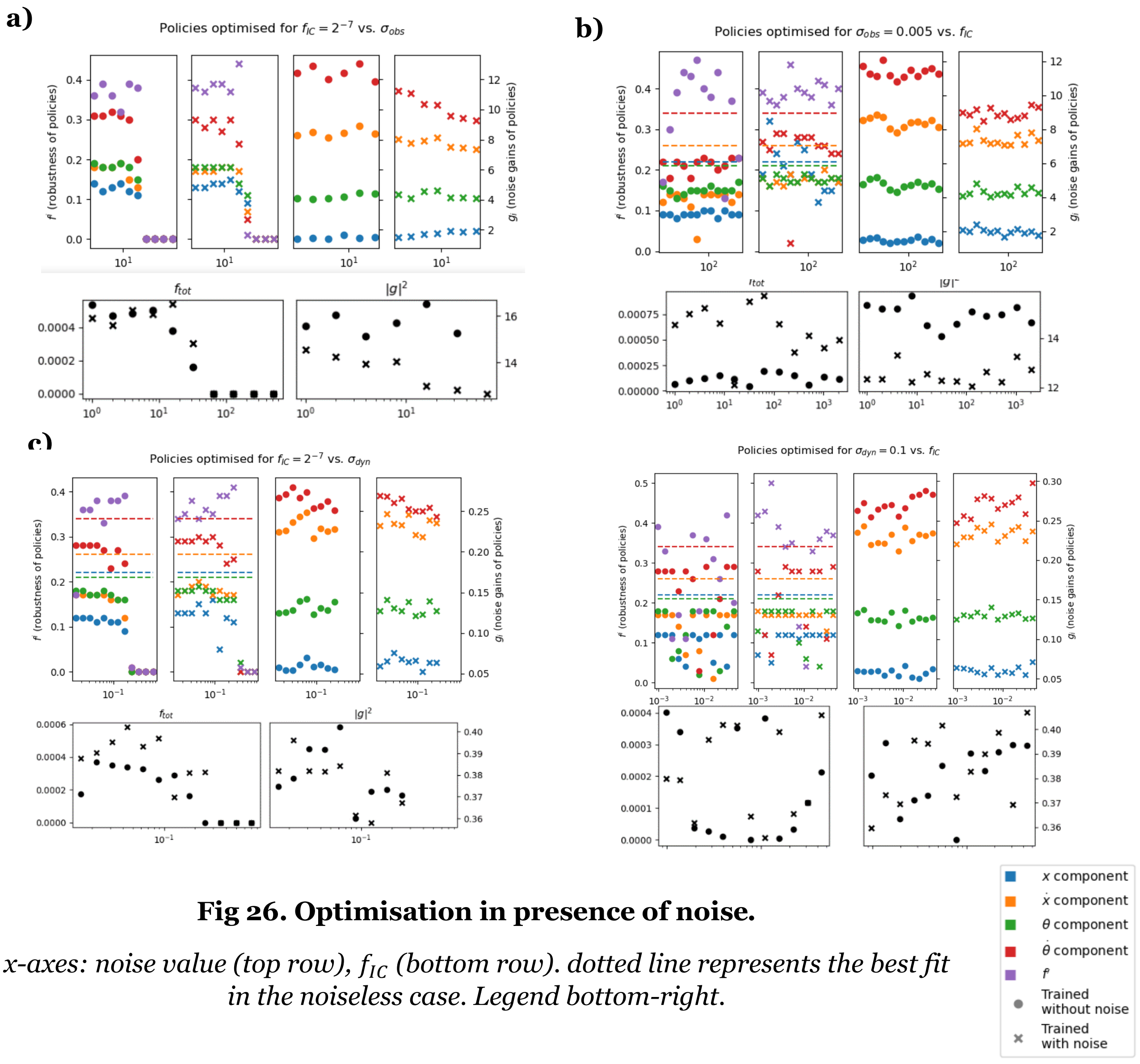


Fig 26. Optimisation in presence of noise.

x-axes: noise value (top row), f_{IC} (bottom row). dotted line represents the best fit in the noiseless case. Legend bottom-right.

prohibitively difficult, at least with these techniques. Differences in the noise types arose in:

- Simulated rollouts of the actual system, where dynamics noise led to divergence from the noiseless trajectory, and therefore likewise in noisy trajectories that were predicted from fitted models.
- Policies optimised in the presence of observation noise perform marginally, but not substantially, better than those optimised in the noiseless case for observed noise. However, for dynamics noise, they performed about the same.
- After adjusting by the factor of 18 to make noise levels analogous, the same level of dynamics noise led to better models but worse controllers.

Thresholds for the recovering noiseless performance model and the policy were similar, so up to these thresholds, model predictive control with an entirely noisy pipeline is still feasible.

Nonlinear Controller.

Outline

A modest number (10-15) of summed Gaussian basis with arbitrary covariances and positions were to be used to create a better policy for the system, of the form

$$p(\boldsymbol{X}) = \sum_i w_i e^{-0.5(\boldsymbol{X}-\boldsymbol{X}_i)^T \boldsymbol{W}(\boldsymbol{X}-\boldsymbol{X}_i)}$$

The controller was to swing the cartpole up to $\boldsymbol{X} = \boldsymbol{0}$ from a downward equilibrium at any initial position or velocity. Ideally it would be able to start from any initial condition, but this problem is harder, and a guaranteed (if slow) way to solve it is by waiting until friction brings the pendulum to the downward equilibrium – turning it into the first problem.

The true, noiseless system dynamics were used for optimisation for simplicity.

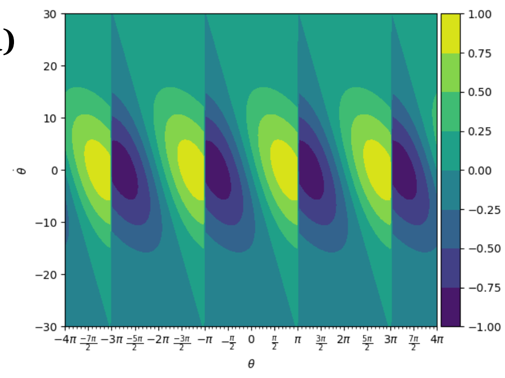
Periodicity

The periodic nature of the θ variable is harder to incorporate into this problem than the previous ones due to the off-diagonal $\boldsymbol{W}$ components. The following options are available:

1. Remapping the θ variable back to the $[-\pi, \pi]$ range. This resulted in discontinuities, particularly bad for basis functions near $\theta = \pi$ (fig. 27a).
2. Directly using a sinusoidal difference for the θ-term between $\boldsymbol{X}$ and $\boldsymbol{X}_i$. Having nonzero off-diagonals causes distortion from intended shapes, but there are no problem for diagonal covariance matrices.
3. Eigen-decomposing $\boldsymbol{W}$ to $\boldsymbol{\Lambda D \Lambda}^{\mathbf{T}}$, where $\boldsymbol{D}$ is diagonal, so that a periodicity can be introduced in the eigenbasis (fig. 27b).

For most attempts at nonlinear policies, a combination of (1) and (2) were used, where basis functions using (1) were kept near $\theta = 0$ to minimise discontinuity. This required having different $\boldsymbol{W}$'s for different basis functions, increasing parameter complexity. (3) was done for final attempts but not enough time was available to functions formulated with the technique fully.

a)



b)

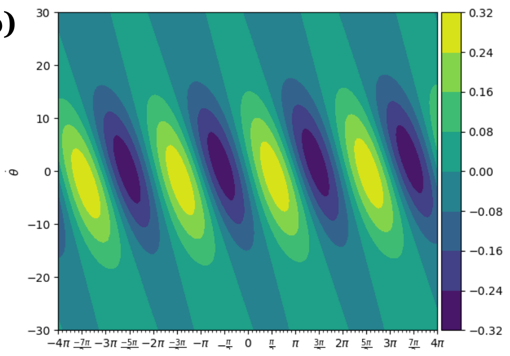


Fig 27. Ways to introduce periodicity. *Techniques are described above.*

Strategy

Immediately it is clear that there are many parameters available, which will affect the model in unclear ways. A key challenge was exploring and parametrising this high-dimensional landscape in a way that allows optimisation or intuition about what kinds of functions might work. A number of initial thoughts were considered:

Simplified problem exploration

The strategy of initially discounting the x and $\dot{x}$ variables, then reintroducing them once working policies were found was very successful for the linear policy. In this problem, finding any working solution is even harder, so this approach is even more valuable.

Experimentation did not reach the point of reintroducing $\dot{x}$ or x due to time constraints.

Types of loss function

The loss function used for the linear policy may not be as effective here. Alternatives and modifications were considered – principally:

1. **Energy-based.**
   From the original system analysis, it is clear that the policy must aim for a state with zero total energy. A loss function $L_e(\boldsymbol{X}) = 1 - e^{-(E(\boldsymbol{X}))^2}$ was created accordingly. Energy does not uniquely define the final system state, so this cannot be expected to fully solve the problem, but it should be helpful for going from rest to a zero-energy trajectory. In a typical reinforcement learning application, insights such as energy may not be available about the system, but there is no reason not to use the information here.
2. **Discount factor.**
   As previously discussed, this should help avoid punishing trajectories that briefly sweep through costly states to reach the target.

(1) was used to good effect. Attempts were made to combine it with the loss function from before and have a single loss function for all cases, but this was not fruitful. (2) was experimented with but did not help – it would likely be more helpful for optimisation if a good, but not optimal, policy had been obtained.

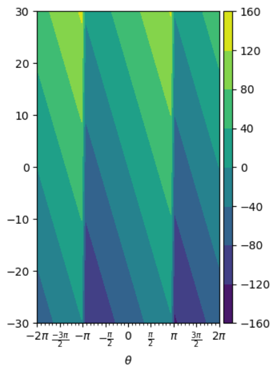
Linear policy

The linear policy was highly robust for its defined basin of convergence. If that basin is reached at any point, this policy could be switched to. However, this seemed against the spirit of the exercise and was not attempted.

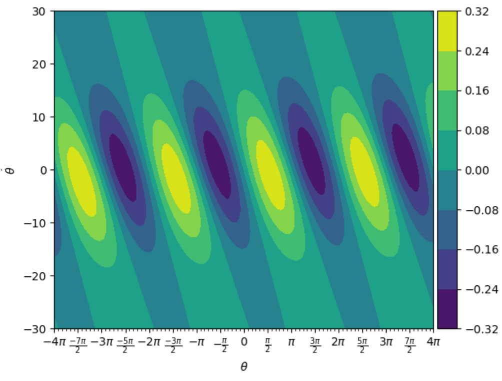
Nonetheless, the linear policy was used to inform experimentation: A similar landscape should be aimed for in the vicinity of the target state. This guided the choice of covariance matrix eigenvectors to be parallel and perpendicular to the optimal $\boldsymbol{p}$ vector, and initial policies were created with pairs of Gaussians that had their centres a distance of twice their $\boldsymbol{p}$-parallel scales apart, in order to obtain a locally linear function equal to that of the linear policy (Fig. 28)

Fig 28. Approximation of linear policy.
a)
Contour plot of optimal policy in $\theta - \dot{\theta}$ plane.
b)
Gaussian pairs that locally approximate the linear policy near the origin.
c)
1d illustration of local linearity of this type of Gaussian pair, with superimposed straight line.
Scales, heights and centres all at 1, line gradient $\sqrt{2}$

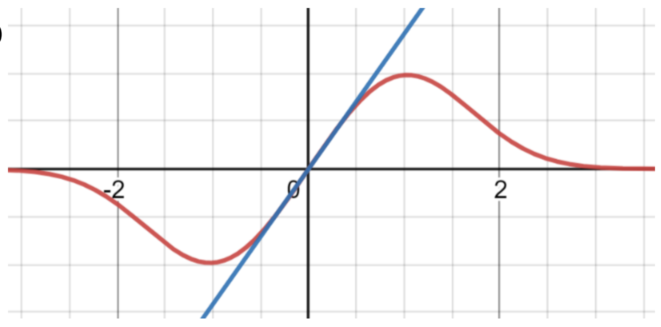
a)



b)



c)



This system was parametrised in terms of the eigendirection scales and optimised with the original loss function. The resulting policy was effective for trajectories starting near the target state, except for oscillations in $\dot{\theta}$ that appeared at convergence for a lot of variables (figure 29). The origin of this effect was not ascertained.

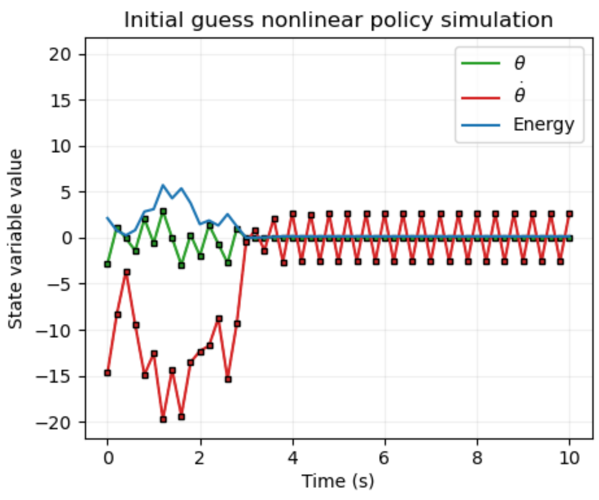


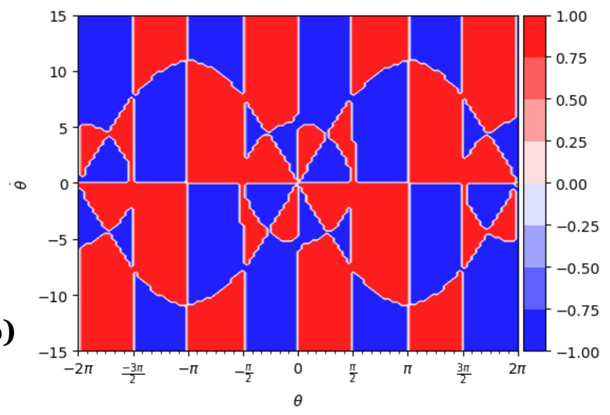
Fig 29. Oscillations at convergence for first attempt at a policy.

Several parametrisations were attempted of arbitrary basis functions were attempted. Initially, a lot of focus was given to ensuring the sign of the force would bring the energy to the required value, and initialised in pairs as in figure 30a. However, it was realised after many failed attempts that the finite number of Euler steps in the simulation makes this landscape force-dependence and far more complicated. This approach was therefore abandoned.

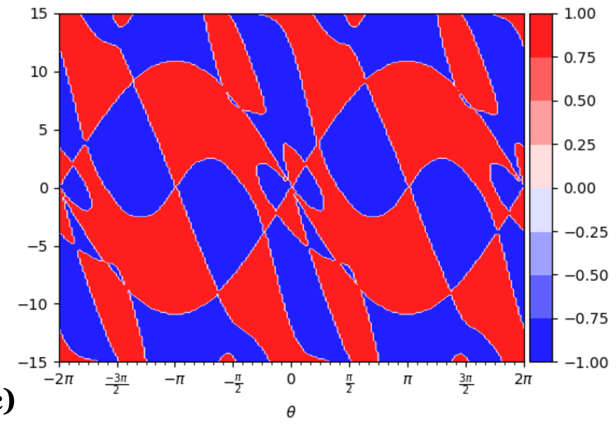
Scatter plots were made superposing states from trajectories on the policy landscape to evaluate policy performance and understand how policies behaved. (figure 31).

To find policies which 'kicked up' the pole to the correct energy, a range of ICs near the downward equilibrium were used, with various parametrisations of basis functions, and the energy-based loss. It was found that parametrisation must be picked carefully to avoid degenerate solutions such as **fig. 32.**

a)



b)



c)

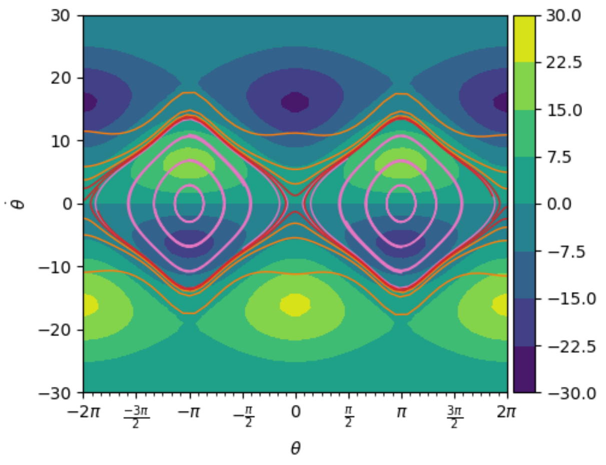


Fig 30. Sign of force required to bring the system to E=0, for an infinitesimal timestep (**a)**) and a finite timestep (**b),** F=20). **c)** Initial naïve attempt at setting basis function locations, with phase space trajectories superimposed.

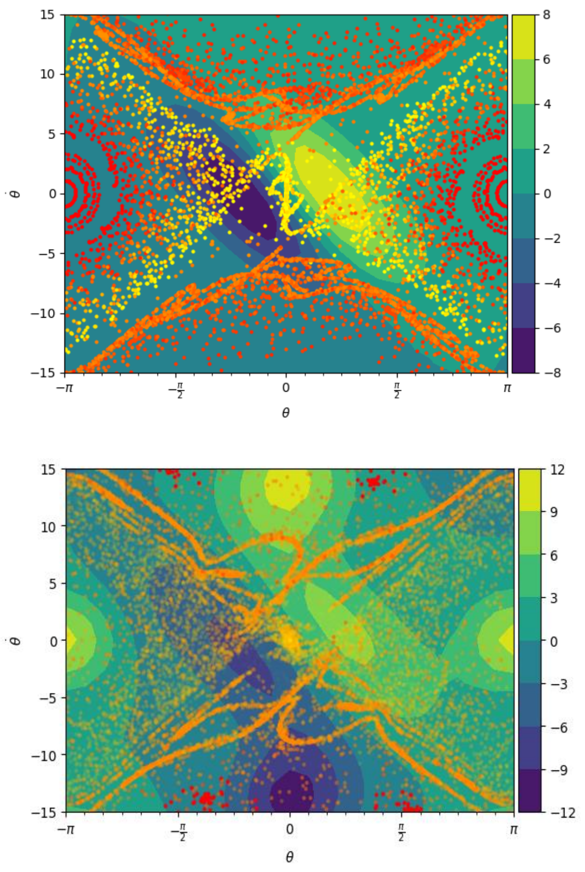


Figure 31. Evaluation of policies.

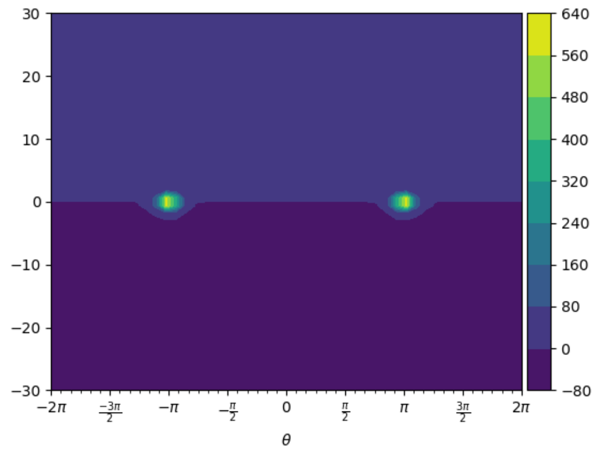


Figure 32. Degenerate solution for kick-up.

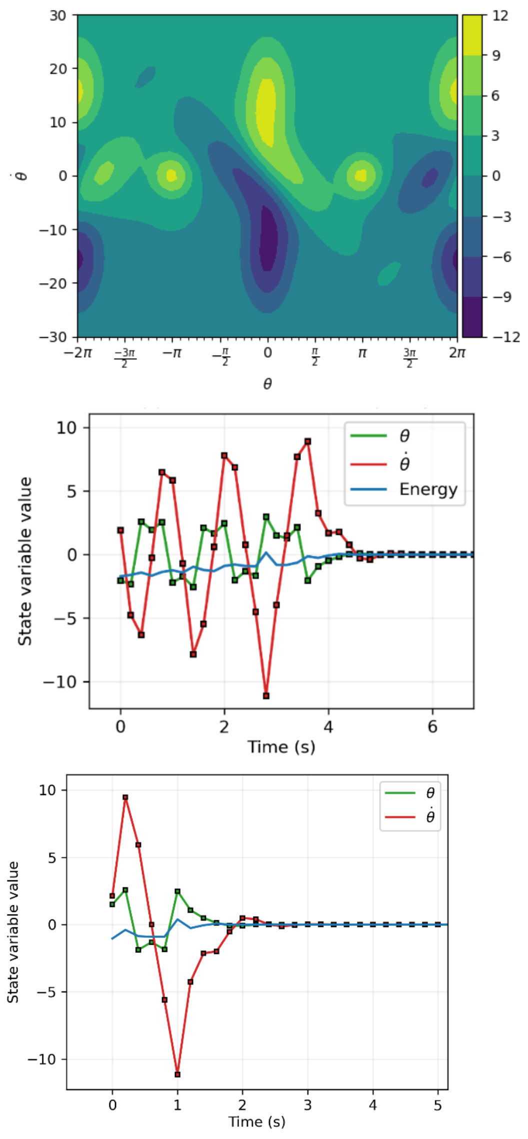


Figure 33. Working policy - its phase space contours and sample trajectories

One optimisation of a kick-up policy with the linear-like controller superimposed resulted in a working policy. It was unstable for energies greater significantly greater than zero (resulting in limit cycles) but converged in almost all cases starting anywhere with energy less than zero.

However, time ran out at this point and no more policies were explored.

Appendix

NOTE – updated equations still missing a factor of L given to gabor. Corrected here, I think.

A1: Energy definition

Let T be kinetic energy, V be potential energy with zero reference at $\theta = 0$, and E be total energy.

$$T = \frac{1}{2}(M+m)\dot{x}^2 + \frac{1}{2}mL\dot{x}\dot{\theta}\cos\theta + \frac{1}{6}mL^2\dot{\theta}^2$$

$$V = \frac{1}{2}mgL(\cos\theta - 1)$$

and $E = T + V$, so

$$E = \frac{1}{2}\dot{x}^2 + \frac{1}{8}\dot{x}\dot{\theta}\cos\theta + \frac{1}{48}\dot{\theta}^2 + \frac{g}{8}(\cos\theta - 1)$$

A2: Equations of motion

Defining $\mu = 4(m+M) - 3m\cos^2\theta$, the equations of motion are

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{1}{\mu}\left(2mL\dot{\theta}^2\sin\theta - 3mg\cos\theta\sin\theta + 4F - 4\mu_x\dot{x} + \frac{6}{L}\mu_\theta\dot{\theta}\cos\theta\right) \\ \dot{\theta} \\ \frac{3}{\mu}\left(-m\dot{\theta}^2\cos\theta\sin\theta + 2\frac{(m+M)}{mL}(mg\sin\theta) - \frac{2}{L}F\cos\theta + \frac{2}{L}\mu_x\dot{x}\cos\theta - 4\frac{(m+M)}{mL^2}\mu_\theta\dot{\theta}\right) \end{bmatrix}$$

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{1}{4-1.5\cos^2\theta}\left(.5\dot{\theta}^2\sin\theta - 1.5g\cos\theta\sin\theta + 0.001(12\dot{\theta}\cos\theta - 4\dot{x}) + 4F\right) \\ \dot{\theta} \\ \frac{3}{4-1.5\cos^2\theta}\left(-.5\dot{\theta}^2\cos\theta\sin\theta + 4g\sin\theta + 0.001(4\dot{x}\cos\theta - 32\dot{\theta}) - 4F\cos\theta\right) \end{bmatrix}$$

A3: Correlation matrix with N=5000

```
[[ 4.08391096e-04  2.00806862e-01  4.18320139e-02  2.62501158e-03]

 [-3.52547831e-03  6.42016490e-03  2.06380111e-01  2.61166600e-02]

 [-1.28462041e-03  1.28783926e-03  1.80439712e-01  1.97702879e-01]

 [-1.32082079e-02  1.29174134e-02  1.25855381e+00 -6.06674820e-02]]
```

A4: Linear policy: attempts at converging from random ICs.

log2 f_IC	f_IC	good loss	loss found	# "successes"
-10.0	0.0010	–11.0862896	-11.203	4
-9.5	0.0014	–10.4142185	-10.989	3
-9.0	0.0020	-9.6572688	-9.842	6
-8.5	0.0028	–8.95858149	-8.94	2
-8.0	0.0039	–8.14534966	-8.133	1
-7.5	0.0055	–7.54007225	–7.675132	3
-7.0	0.0078	–6.85972825	-0.658	0
-6.5	0.0110	-6.2141508	-6.247	1
-6.0	0.0156	–5.56545283	-0.348	0
-5.5	0.0221	–4.82166166	-0.346	0
-5.0	0.0313	–4.39724891	-4.226	1
-4.5	0.0442	–3.53923919	-0.638	0
-4.0	0.0625	–3.03818484	-0.665	0
-3.5	0.0884	–2.41043892	-0.589	0
-3.0	0.1250	–1.96359157	-0.629	0
-2.5	0.1768	–1.47152983	-0.599	0
-2.0	0.2500	–0.98561673	-0.603	0
-1.5	0.3536	–0.69776739	-0.571	0
-1.0	0.5000	–0.44274501	-0.58	0
-0.5	0.7071	–0.34012099	-0.598	0
0.0	1.0000	–0.26251975	-0.538	0