ACCELERATING SIMULATION OF STIFF NONLINEAR SYSTEMS USING CONTINUOUS TIME ECHO STATE NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern design, control, and optimization often require multiple expensive simulations of highly nonlinear stiff models. These costs can be amortized by training a cheap surrogate of the full model, which can then be used repeatedly. Here we present a general data-driven method, the continuous-time echo state network (CTESN), for generating surrogates of nonlinear ordinary differential equations with dynamics at widely separated timescales. We showcase the ability for this surrogate to accurately handle highly stiff systems which have been shown to cause training failures with common surrogate methods such as Physics-Informed Neural Networks (PINNs), Long Short Term Memory (LSTM) networks, and discrete echo state networks (ESN). We show that our model captures fast transients as well as slow dynamics, while demonstrating that fixed time step machine learning techniques are unable to adequately capture the multi-rate behavior. This provides compelling evidence for the ability of CTESN surrogates to predict and accelerate highly stiff dynamical systems which are unable to be directly handled by previous scientific machine learning techniques.

1 INTRODUCTION

Stiff nonlinear systems of ordinary differential equations are widely prevalent throughout science and engineering (Wanner & Hairer, 1996; Shampine & Gear, 1979) and are characterized by dynamics with widely separated time scales. These systems require highly stable numerical methods to use non-vanishing step-sizes reliably gear1971numerical, and also tend to be computationally expensive to solve. Even with state-of-the-art simulation techniques, design, control, and optimisation of these systems remains intractable in many realistic engineering applications (Benner et al., 2015). To address these challenges, researchers have focused on techniques to obtain an approximation to a system (called a "surrogate") whose forward simulation time is relatively inexpensive while maintaining reasonable accuracy (Willard et al., 2020; Ratnaswamy et al., 2019; Zhang et al., 2020; Kim et al., 2020; van de Plassche et al., 2020).

A popular class of traditional surrogatization techniques is projection based model order reduction, such as the proper orthogonal decomposition (POD) (Benner et al., 2015). This method computes "snapshots" of the trajectory and uses the singular value decomposition of the linearization in order to construct a basis of a subspace of the snapshot space, and the model is remade with a change of basis. However, if the system is very nonlinear, the computational complexity of this linearization-based reduced model can be almost as high as the original model. One way to overcome this difficulty is through empirical interpolation methods (Nguyen et al., 2014). Other methods to produce surrogates generally utilize the structural information known about highly regular systems like partial differential equation discretizations (Frangos et al., 2010).

Many of these methods require a scientist to actively make choices about the approximations being performed to the system. In contrast, the data-driven approaches like Physics-Informed Neural Networks (PINNs)(Raissi et al., 2019) and Long Short Term Memory (LSTM) networks (Chattopadhyay et al., 2020) have gained popularity due to their apparent applicability to "all" ordinary and partial differential equations in a single automated form. However, numerical stiffness (Söderlind et al., 2015) and multiscale dynamics represent an additional challenge. Highly stiff differential

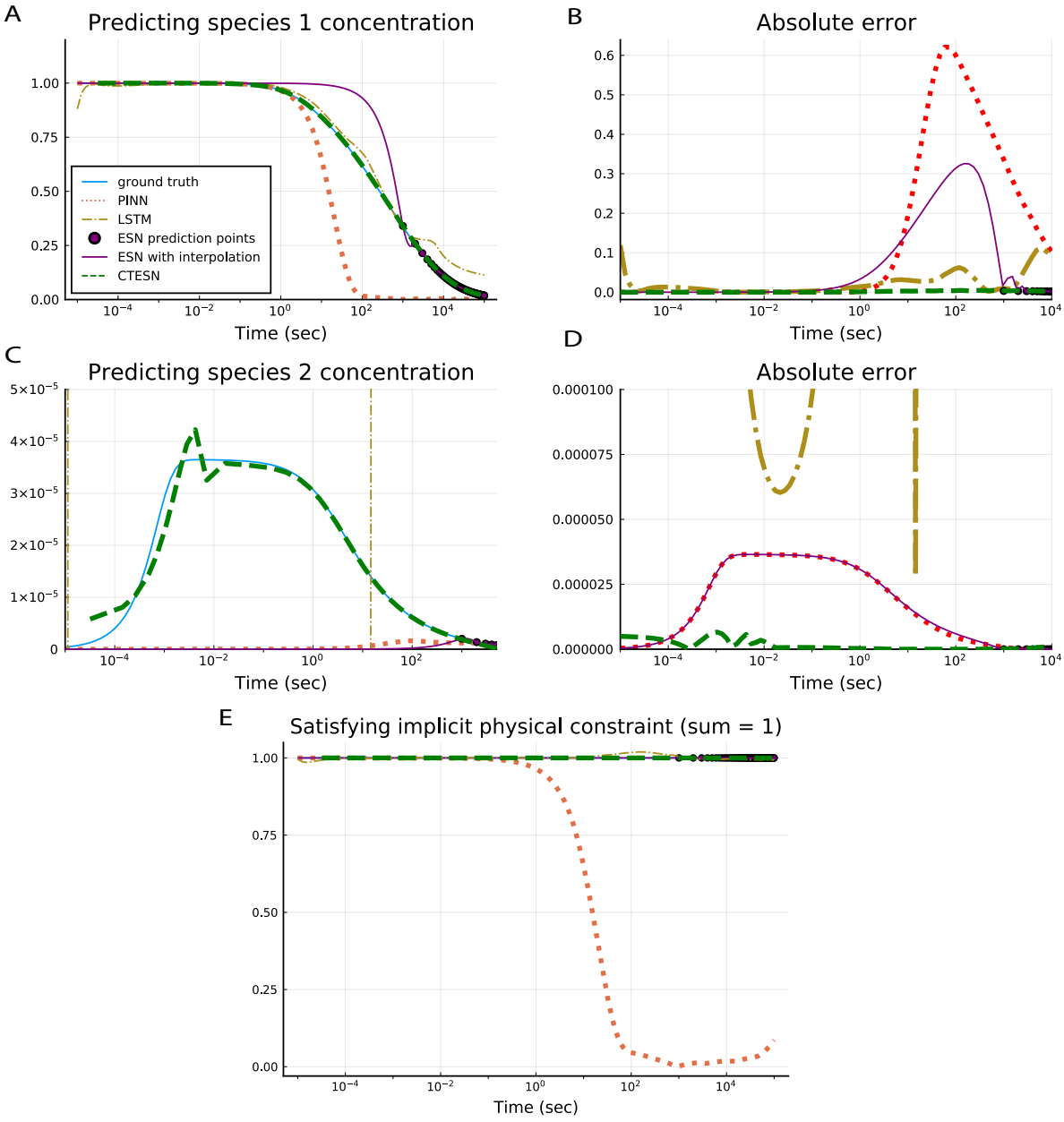

Figure 1: **Prediction of each surrogate on the Robertson's equations** Shown in each figure is the result of the data-driven algorithm's prediction at $p = [0.04, 3 \times 10^7, 1 \times 10^4]$, a parameter set not in the training data. Ground truth, obtained by solving the ODE using the Rosenbrock23 solver with absolute tolerance of 10^{-6}, is in blue. The PINN was trained using a 3-layer multi-layer perceptron with the ADAM optimizer for 300,000 epochs with minibatching, and its prediction is in red. Both the ESN and CTESN were trained with a reservoir size of 3000 on a parameter space of $[0.036, 0.044] \times [2.7 \times 10^7, 3.3 \times 10^7] \times [9 \times 10^3, 1.1 \times 10^4]$, from which 1000 sets of parameters were sampled using Sobol sampling. The predictions of the CTSEN are generated by the radial basis function prediction of $W_{out}(p)$ and are shown in green. Predictions from the ESN are in purple. The LSTM predictions, in gold, are generated by a network with 3 hidden LSTM layers and an output dense layer, after training for 2000 epochs. (A) A timeseries plot of the $y_1(t)$ predictions. (B) The absolute error of the surrogate predictions on $y_1(t)$. (C) A timeseries plot of the $y_2(t)$ predictions. (D) The absolute error of the surrogate predictions on $y_2(t)$. (E) The result of $y_1(t) + y_2(t) + y_3(t)$ over time. By construction this quantity's theoretical value is 1 over the timeseries.

equations can lead to gradient pathologies that make common surrogate techniques like PINNs hard to train (Wang et al., 2020).

The purpose of this work is to introduce a general data-driven method, the CTESN, that is generally applicable and able to accurately capture highly nonlinear heterogeneous stiff time-dependent systems without requiring the user to train on non-stiff approximations. It is able to accurately train and predict on highly ill-conditioned models. We demonstrate these results (Figure 1) on the Roberston's equations, which PINNs, LSTM networks and discrete-time machine learning techniques fail to handle. Our results showcase the ability to transform difficult stiff equations into non-stiff reservoir equations which are then integrated in place of the original system.

2 CONTINUOUS-TIME ECHO STATE NETWORKS

Echo State Networks (ESNs) are a reservoir computing framework which projects signals from higher dimensional spaces defined by the dynamics of a fixed non-linear system called a "reservoir" (Ozturk et al., 2007). The ESN's mathematical formulation is as follows. For a N_R-dimensional reservoir, the reservoir equation is given by:

$$r_{n+1} = f(Ar_n + W_{fb}x_n), \tag{1}$$

where f is a chosen activation function (like tanh or sigmoid), A is the $N_R \times N_R$ reservoir weight matrix, and W_{fb} is the $N_R \times N$ feedback matrix where N is the size of our original model. In order to arrive at a prediction of our original model, we take a projection of the reservoir:

$$\hat{x}_n = g(W_{out}r_n), \tag{2}$$

where g is the output activation function (generally the identity or sigmoid) and W_{out} is the $N \times N_R$ projection matrix. In the training process of an ESN, the matrices A and W_{fb} are randomly chosen constants, meaning the W_{out} matrix is the only variable which needs to be approximated. W_{out} is calculated by using a least squares fit of against the model's time series, which then fully describes the prediction process.

This ability to handle problems with gradient pathologies gives the intuitive justification for exploring reservoir computing techniques on handling stiff equations. However, stiff systems generally have behavior spanning multiple timescales which are difficult to represent with uniformly-spaced prediction intervals. To solve these issues, we introduce a new variant of ESNs, which we call continuous-time echo state networks (CTESNs), which allows for an underlying adaptive time process while avoiding gradient pathologies in training. Let N be the dimension of our model, and let P be a Cartesian space of parameters under which the model is expected to operate. The CTESN of with reservoir dimension N_R is defined as

$$r' = f(Ar + W_{hyb}x(p^*, t)), \tag{3}$$

$$x(t) = g(W_{out}r(t)), \tag{4}$$

where A is a fixed sparse random matrix of dimension $N_R \times N_R$ and W_{hyb} is a fixed random dense matrix of dimensions $N_R \times N$. The term $W_{hyb}x(p^*, t)$ represents a "hybrid" term that incorporates physics information into the reservoir (Pathak et al., 2018), namely a solution at some point in the parameter space of the dynamical system. Given these fixed values, the readout matrix W_{out} is the only trained portion of this network and is obtained through a least squares fit of the reservoir ODE solution against the original timeseries. We note that in this study we choose $f = \tanh$ and $g = \mathrm{id}$ for all of our examples.

To obtain a surrogate that predicts the dynamics at new physical parameters, the reservoir projection W_{out} is fit against many solutions at parameters $\{p_1, \ldots, p_n\}$, where n is the number of training data points sampled from the parameter space. Using these fits, an interpolating function $W_{out}(p)$ between the matrices can be trained. A prediction $\hat{x}(t)$ for at physical parameters $\hat{p}$ is thus given by:

$$\hat{x}(t) = W_{out}(\hat{p})r(t). \tag{5}$$

A strong advantage of our method is its ease of implementation and ease of training. Global L_2 fitting via stabilized methods like SVD are robust to ill-conditioning, alleviating many of the issues encountered when attempting to build neural surrogates of such equations. Also note that in this

particular case, the readout matrix is fit against the same reservoir time series. This means that prediction does not need to simulate the reservoir, providing an extra acceleration.

Another advantage is the ability to use time stepping information from the solver during training. As noted before, not only are step sizes chosen adaptively based on minimizing a local error estimate to a specified tolerance (Shampine & Gear, 1979), but they also adapt to concentrate around the most stiff and numerically difficult time points of the model by incorporating the Newton convergence into the rejection framework. These timestamps thus provide heuristic snapshots of the most important points for training the least squares fit, whereas snapshots from uniform time steps may skip over many crucial aspects of the dynamics.

3 Case Study: Robertson's Equations and High Stiffness

We first consider Robertson's chemical reactions involving three reaction species A, B and C:

$$\begin{aligned} A &\xrightarrow{0.04} B \\ B + B &\xrightarrow{3\times10^7} C + B \\ B + C &\xrightarrow{10^4} A + C \end{aligned}$$

which lead to the ordinary differential equations:

$$\dot{y_1} = -0.04y_1 + 10^4 y_2 \cdot y_3 \tag{6}$$

$$\dot{y_2} = 0.04y_1 - 10^4 y_2 \cdot y_3 - 3 \cdot 10^7 y_2^2 \tag{7}$$

$$\dot{y_3} = 3 \cdot 10^7 y_2^2 \tag{8}$$

where y_1, y_2, and y_3 are the concentrations of A, B and C respectively. This system has widely separated reaction rates $(0.04, 10^4, 3 \cdot 10^7)$, and is well known to be very stiff (Gobbert, 1996; Robertson & Williams, 1975; Robertson, 1976). It is commonly used as an example to evaluate integrators of stiff ODEs (Hosea & Shampine, 1996). Finding an accurate surrogate for this system is difficult because it needs to capture both the stable slow reacting system and the fast transients. Additionally, the surrogate needs to be consistent with this system's implicit physical constraints, such as the conservation of matter ($y_1 + y_2 + y_3 = 1$) and positivity of the variables ($y_i > 0$), in order to provide a stable solution.

The CTESN method is able to accurately capture both the slow and fast transients and respect the conservation of mass. The ESN is able to accurately predict at the time points it was trained on, but many features are missed. The uniform stepping completely misses the fast transient rise at $t = 10^{-4}$ because the uniform intervals do not sample points from that time scale. Additionally, the first sampled time point at $t = 100$ is far into the concentration drop of y_1 which leads to an inaccurate prediction before the system settles into near steady state behavior. As stated earlier, the CTESN uses information from a stiff ODE solver to choose the right points along the time span to accurately capture multi-scale behaviour with less training data than the ESN. In order to compare the discrete ESN to the continuous result, a cubic spline was fit to its 101 evenly spaced prediction points.

Figure 1 highlights how the trained PINN fails to capture both the fast and the slow transients and do not respect mass conservation. Our collaborators investigated why PINNs fail to solve these equations in (Ji et al., 2020). The reason for the difficulty can be attributed to recently identified results in gradient pathologies in the training arising from stiffness (Wang et al., 2020). With a highly ill-conditioned Hessian in the training process due to the stiffness of the equation, it is very unlikely for local optimization to find a parameters which make an accurate prediction. We additionally note stiff systems of this form may be hard to capture by neural networks directly as neural networks show a bias towards low frequency functions (Rahaman et al., 2019).

4 CONCLUSION

We present CTESNs, a data-driven method for generating surrogates of nonlinear ordinary differential equations with dynamics at widely separated timescales. Our method maintains accuracy for different parameters in a chosen parameter space, and shows favourable scaling with system size on a physics-inspired scalable model. This method can be applied to any ordinary differential equation without requiring the scientist to simplify the model before surrogate application, greatly improving productivity.

ACKNOWLEDGEMENT

The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Numbers DE-AR0001222 and DE-AR0001211, and NSF grant OAC-1835443. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. The authors thank Francesco Martinuzzi for reviewing drafts of this paper.

REFERENCES

Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

Francesco Casella. Simulation of large-scale models in modelica: State of the art and future perspectives. In *Proceedings of the 11th International Modelica Conference*, pp. 459–468, 2015.

Ashesh Chattopadhyay, Pedram Hassanzadeh, and Devika Subramanian. Data-driven predictions of a multiscale lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, 27(3):373–389, 2020.

Michalis Frangos, Youssef Marzouk, Karen Willcox, and B van Bloemen Waanders. Surrogate and reduced-order modeling: a comparison of approaches for large-scale statistical inverse problems [chapter 7]. 2010.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Matthias K Gobbert. Robertson's example for stiff differential equations. *Arizona State University, Technical report*, 1996.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

ME Hosea and LF Shampine. Analysis and implementation of tr-bdf2. *Applied Numerical Mathematics*, 20(1-2):21–37, 1996.

Weiqi Ji, Weilun Qiu, Zhiyu Shi, Shaowu Pan, and Sili Deng. Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. *arXiv preprint arXiv:2011.04520*, 2020.

Youngkyu Kim, Youngsoo Choi, David Widemann, and Tarek Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *arXiv preprint arXiv:2009.11990*, 2020.

Diederik P Kingma and J Adam Ba. A method for stochastic optimization. arxiv 2014. *arXiv preprint arXiv:1412.6980*, 434, 2019.

VB Nguyen, M Buffoni, K Willcox, and BC Khoo. Model reduction for reacting flow applications. *International Journal of Computational Fluid Dynamics*, 28(3-4):91–105, 2014.

Mustafa C Ozturk, Dongming Xu, and José C Prıncipe. Analysis and design of echo state networks. *Neural computation*, 19(1):111–138, 2007.

Jaideep Pathak, Alexander Wikner, Rebeckah Fussell, Sarthak Chandra, Brian R Hunt, Michelle Girvan, and Edward Ott. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(4):041101, 2018.

Christopher Rackauckas and Qing Nie. Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Akshay Ranade and Francesco Casella. Multi-rate integration algorithms: a path towards efficient simulation of object-oriented models of very large systems. In *Proceedings of the 6th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pp. 79–82, 2014.

Vishagan Ratnaswamy, Cosmin Safta, Khachik Sargsyan, and Daniel M Ricciuto. Physics-informed recurrent neural network surrogates for e3sm land model. *AGUFM*, 2019:GC43D–1365, 2019.

HH Robertson. Numerical integration of systems of stiff ordinary differential equations with special structure. *IMA Journal of Applied Mathematics*, 18(2):249–263, 1976.

HH Robertson and J Williams. Some properties of algorithms for stiff differential equations. *IMA Journal of Applied Mathematics*, 16(1):23–34, 1975.

Lawrence F Shampine. Implementation of rosenbrock methods. *ACM Transactions on Mathematical Software (TOMS)*, 8(2):93–113, 1982.

Lawrence F Shampine and Charles William Gear. A user's view of solving stiff ordinary differential equations. *SIAM review*, 21(1):1–17, 1979.

Ilya M Sobol', Danil Asotsky, Alexander Kreinin, and Sergei Kucherenko. Construction and comparison of high-dimensional sobol'generators. *Wilmott*, 2011(56):64–79, 2011.

Gustaf Söderlind, Laurent Jay, and Manuel Calvo. Stiffness 1952–2012: Sixty years in search of a definition. *BIT Numerical Mathematics*, 55(2):531–558, 2015.

KL van de Plassche, J Citrin, C Bourdelle, Y Camenen, FJ Casson, F Felici, P Horn, A Ho, S Van Mulders, F Koechl, et al. Fast surrogate modelling of turbulent transport in fusion plasmas with physics-informed neural networks. 2020.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv preprint arXiv:2001.04536*, 2020.

Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations II*. Springer Berlin Heidelberg, 1996.

Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 2020.

Ruixi Zhang, Remmy Zen, Jifang Xing, Dewa Made Sri Arsa, Abhishek Saha, and Stéphane Bressan. Hydrological process surrogate modelling and simulation with neural networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 449–461. Springer, 2020.

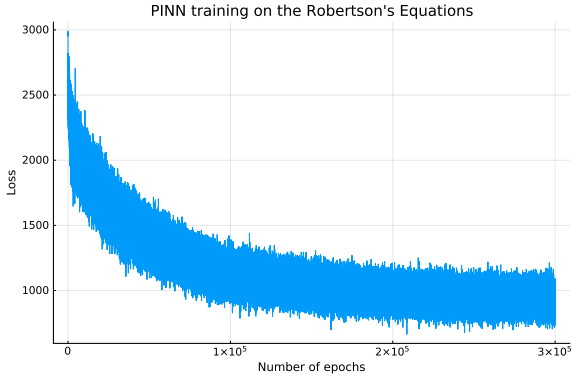

Figure 2: **Training a PINN on the Robertson's Equations**: PINN was trained for 300,000 epochs using the ADAM optimizer with a learning rate of 10^{-3}, by which time the loss seems to saturate. The hyperparameters of the PINN can be found in the Case Studies section.

A Appendix

A.1 Fitting Robertson's Equations - Hyperparameters

A surrogate was trained by sampling 1000 sets of parameters from the Cartesian parameter space $[0.036, 0.044] \times [2.7 \times 10^7, 3.3 \times 10^7] \times [9 \times 10^3, 1.1 \times 10^4]$ using Sobol sampling so as to evenly cover the whole space. We train on the time series of the three states themselves as outputs. A least squares projection W_{out} was fit for each set of parameters, and then a radial basis function was used to interpolate between the matrices. The prediction workflow is as follows: given a set of parameters whose time series is desired, the radial basis function predicts the projection matrix. The pre-simulated reservoir is then sampled at the desired time points, and a matrix multiplication with the predicted W_{out} gives us the desired prediction. Figure 1 shows a comparison between the CTESN, ESN, PINN and LSTM methods. The PINN data is reproduced from (Ji et al., 2020) and the ESN was trained using 101 time points uniformly sampled from the time span, while CTESN used 61 adaptively sampled time points informed by the ODE solver (Rosenbrock23 (Shampine, 1982)).

The PINN was trained by sampling 2500 logarithmically spaced points across the time span. The network used was a 3-layer multi-layer perceptron with 128 nodes per hidden layer and the Gaussian Error Linear Unit activation function (Hendrycks & Gimpel, 2016). The layers were initialed using Xavier initialization (Glorot & Bengio, 2010), and trained with the ADAM optimizer (Kingma & Ba, 2019) at a learning rate of 10^{-3} for 300,000 epochs with mini batch size of 128. Figure 2 shows the convergence plot as the PINN trains on the ROBER equations. The LSTM network used a similar architecture to the PINN, but with LSTM hidden layers instead of fully connected layers. It used 2500 logarithmically spaced points and was trained for 2000 epochs until convergence.

A.2 Practical implementation

In this section we describe the automated training procedure used to generate CTESN surrogates. An input parameter space P is first chosen. This could be a design space for the model or a range of operating conditions. Now n sets of parameters $\{p_1, \ldots, p_n\}$ are sampled from this space using a sampling sequence that covers as much of the space as possible. The full model is now simulated at each sample in parallel since each run is independent, generating time series for each run. The choice of points in time used to generate the time series at each p comes from the numerical ODE solve at that p. The reservoir ODE is then constructed using a candidate solution at any one of the n parameters $x(p^*, t), p^* \in \{p_1, \ldots, p_n\}$ and is then simulated, generating the reservoir time series. Since the reservoir ODE is non-stiff, this simulation is cheap compared to the cost of the full model. Least squares projections can now be calculated from each solution to the reservoir in parallel. Once all the least squares matrices are obtained, an interpolating function is trained to predict the least

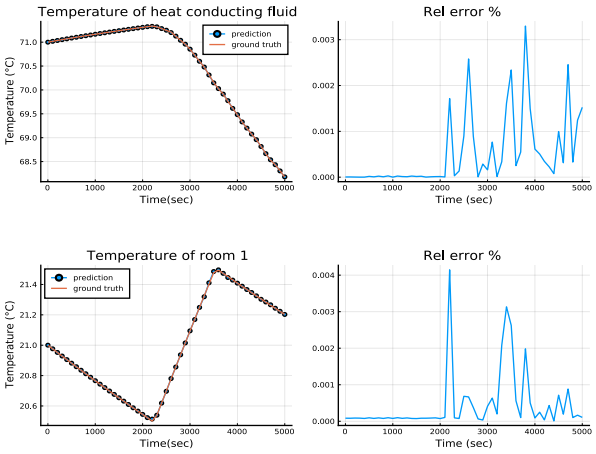

Figure 3: **Validating the surrogate of the scalable heating system with 10 rooms.** When tested with parameters it has not seen in training, our surrogate is able to reproduce the behaviour of the system to within 0.01% error. The surrogate is trained on 100 points sampled from the $[17C, 23C] \times [65C, 75C]$ where the ranges represent set point temperature of each room and set point of the fluid supplying heat to the rooms respectively. The test parameters that validated here are $[21C, 71C]$. More details on training can be found in the Case Studies section.

squares projection matrix. Both the least squares fitting and training the interpolating function are, in practice, much cheaper than the cost of simulating the model multiple times.

Prediction comprises of two steps: predicting the least squares matrix, and simulating the reservoir time series (or, in this case, just using the pre-computed continuous solution since the reservoir is fixed for every set of parameters). The final prediction is just the matrix multiplication of two.

A strong advantage of the training is that it requires no manual investigation of the stiff model on the part of the researcher and can be called as an off-the-shelf routine. It allows the researcher to make a trade-off, computing a few parallelized runs of the full stiff model in order to generate a surrogate, which can then be plugged in and used repeatedly for design and optimization.

We implemented the training routines and the following models in the Julia programming language (Bezanson et al., 2017) to take advantage of its first class support for differential equations solvers (Rackauckas & Nie, 2017) and scientific machine learning packages. For the examples in this paper, we have sampled the high-dimensional spaces using a Sobol low-discrepancy sequence (Sobol' et al., 2011) and interpolated the W_{out} matrices using a radial basis function provided by the Julia package Surrogates.jl (`https://github.com/SciML/Surrogates.jl`).

A.3 Stiffly-Aware Surrogates of HVAC Systems

We also present a scalable benchmark used in the engineering community (Casella, 2015). It is a simplified, lumped-parameter model of a heating system with a central heater supplying heat to several rooms through a distribution network. Each room has an on-off controller with hysteresis which provides very fast localized action (Ranade & Casella, 2014). The resulting system of equations is thus very stiff and unable to be solved by standard explicit time stepping methods.

The size of the heating system is scaled by a parameter N which refers to the number of users/rooms. Each room is governed by two equations corresponding to its temperature and the state of its on-off controller. The temperature of fluid supplying heat to each room is governed by one equation. This produces a system with $2N + 1$ coupled non-linear equations. This "scalability" lets us test how our CTESN surrogate scales. To train the surrogate, we define a parameter space P under which we expect it to operate. First, we assume set point temperature of each room to be between $17C$ and $23C$. Each room is warmed by a heat conducting fluid, whose set point is between $65C$ and

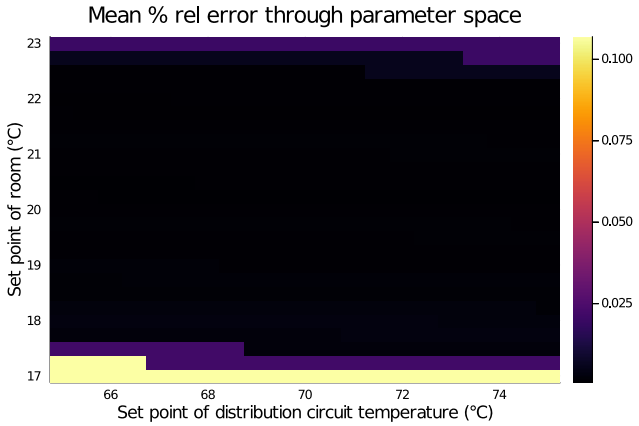

Figure 4: **Reliability of surrogate through parameter space.** We sampled our grid at over 500 grid points and plotted a heatmap of test error through our parameter space. We find our surrogate performs reliably even at the border of our space with error within 0.1%

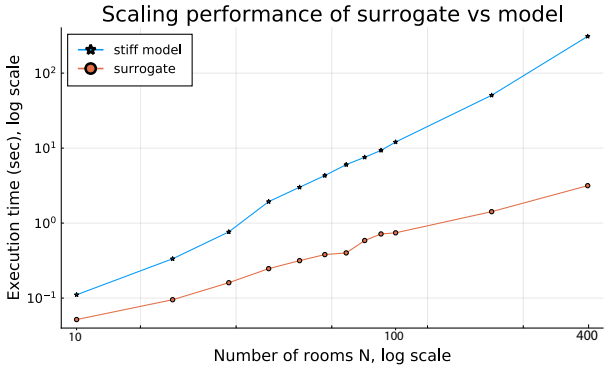

Figure 5: **Scaling performance of surrogate on heating system.** We compare the time taken to simulate the full stiff model to the trained surrogate with 10, 20, 30 , 40, 50, 60, 70, 80, 90, 100, 200 and 400 rooms. We observe a speedup of up to 98x. The surrogate was trained by sampling 100 sets of parameters from our input space, with a reservoir size of 3000.

$75C$. Thus the parameter space over which we expect our surrogate to work is the rectangular space denoted by $[17C, 23C] \times [65C, 75C]$.

We used a reservoir size of 3000 and sampled 100 sets of parameters from this space using Sobol sampling, and fit least squares projection matrices W_{out} between each solution and the reservoir. For a system with N rooms, we train on $N + 1$ outputs, namely the temperature of each room, and the temperature of the heat conducting fluid. Figure 3 demonstrates that the training technique is accurately able to find matrices W_{out} which capture the stiff system within 0.01% error on a test parameters. We then fit an interpolating radial basis function $W_{out}(p)$. Figure 4 demonstrates that the interpolated $W_{out}(p)$ is able to adequately capture the dynamics throughout the trained parameter space. Lastly, Figure 5 demonstrates the $O(N)$ cost of the surrogate evaluation, which in comparison to the $O(N^3)$ cost of a general implicit ODE solver (due to the LU-factorizations) leads to an increasing gap in the solver performance as N increases. At the high end of our test, the surrogate accelerates a 801 dimensional stiff ODE system by approximately 98x.