A Mesh-Free PINN Framework for Scalable Stellar Structure Modeling

Philipp M. Srivastava

Northwestern University

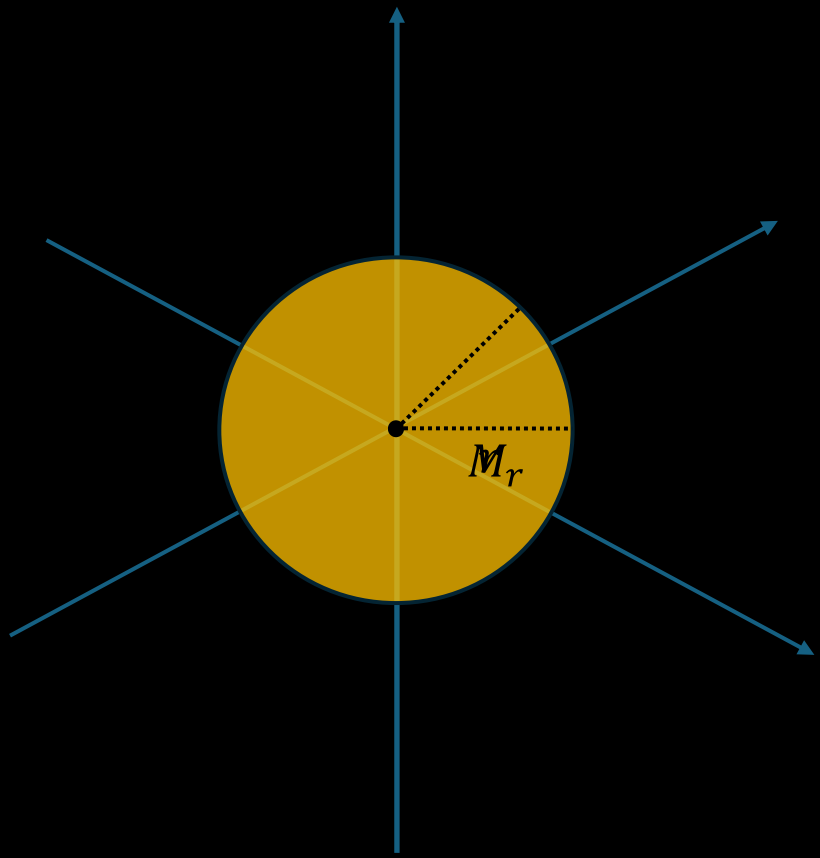
Work by: Manuel Ballester, Santiago Lopez-Tapia, Seth Gossage, Patrick Koller, Philipp M. Srivastava, Ugur Demir, Yongseok Jo, Almudena P. Marquez, Christoph Wuersch, Souvik Chakraborty, Vicky Kalogera, and Aggelos Katsaggelos

ArXiv:2604.06255

Cosmic Horizons Conference
2026-07-14

Funded by the U.S. National Science Foundation and the Simons Foundation

Defining Stellar Structure



- Consider temperature varying from the star's center to its surface
- Assume temperature changes equally in all radial directions
- Instead of using r as a spatial coordinate, we consider the mass enclosed

<u>Stellar Structure Quantities</u>

Pressure – $P(M_r)$

Luminosity – $L_r(M_r)$

Radius – $r(M_r)$

Temperature – $T(M_r)$

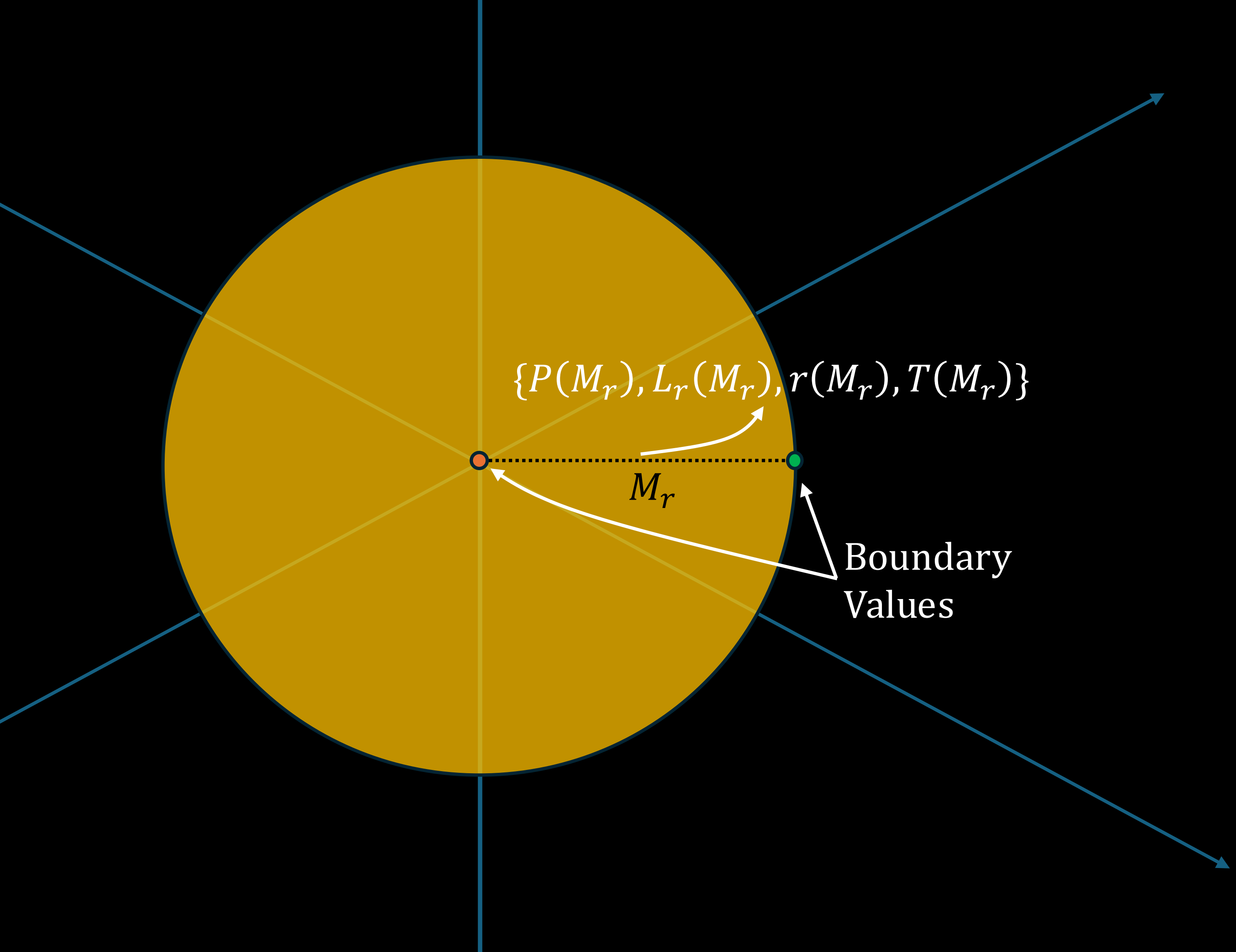
<u>Auxiliary Quantities</u>

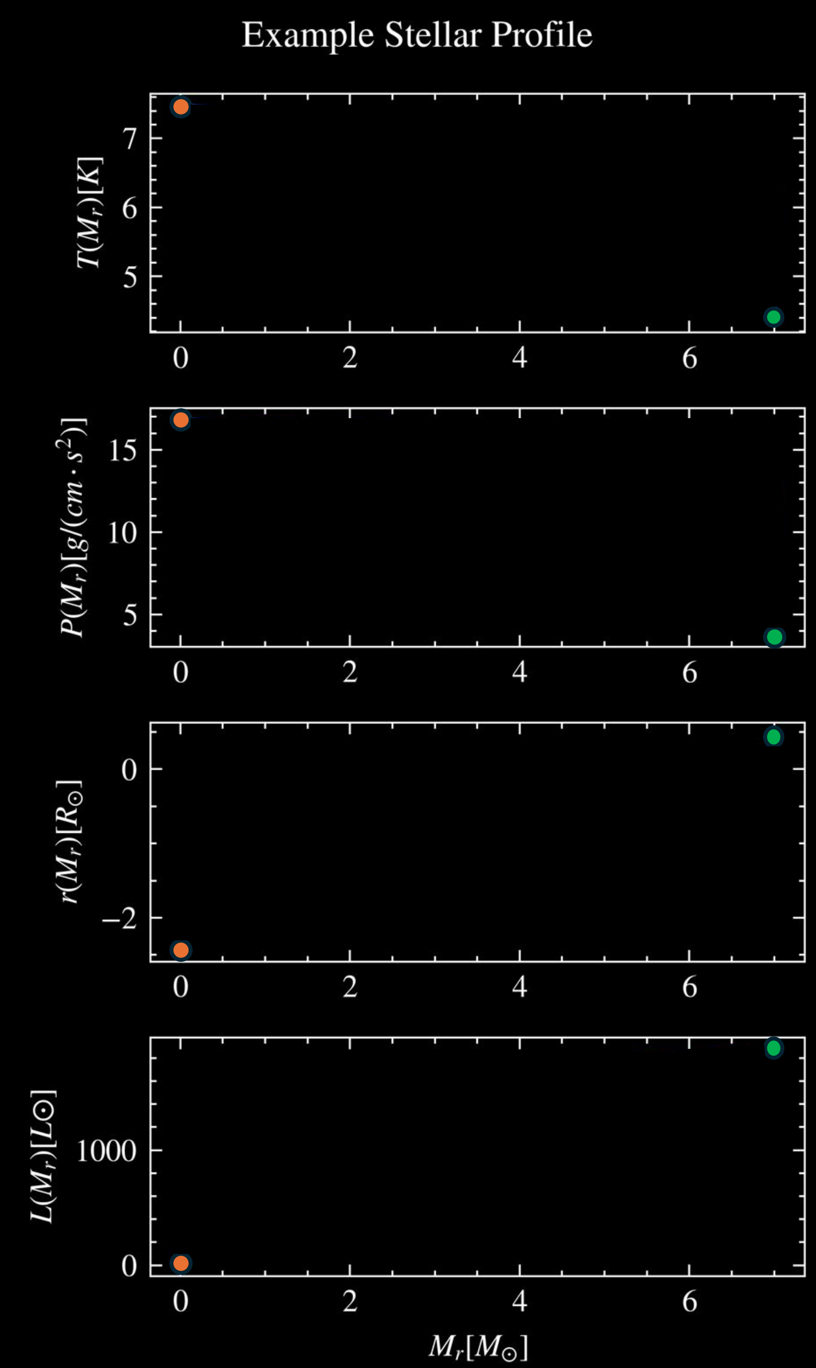
Density – $\rho(M_r)$

Opacity - $\kappa(M_r)$

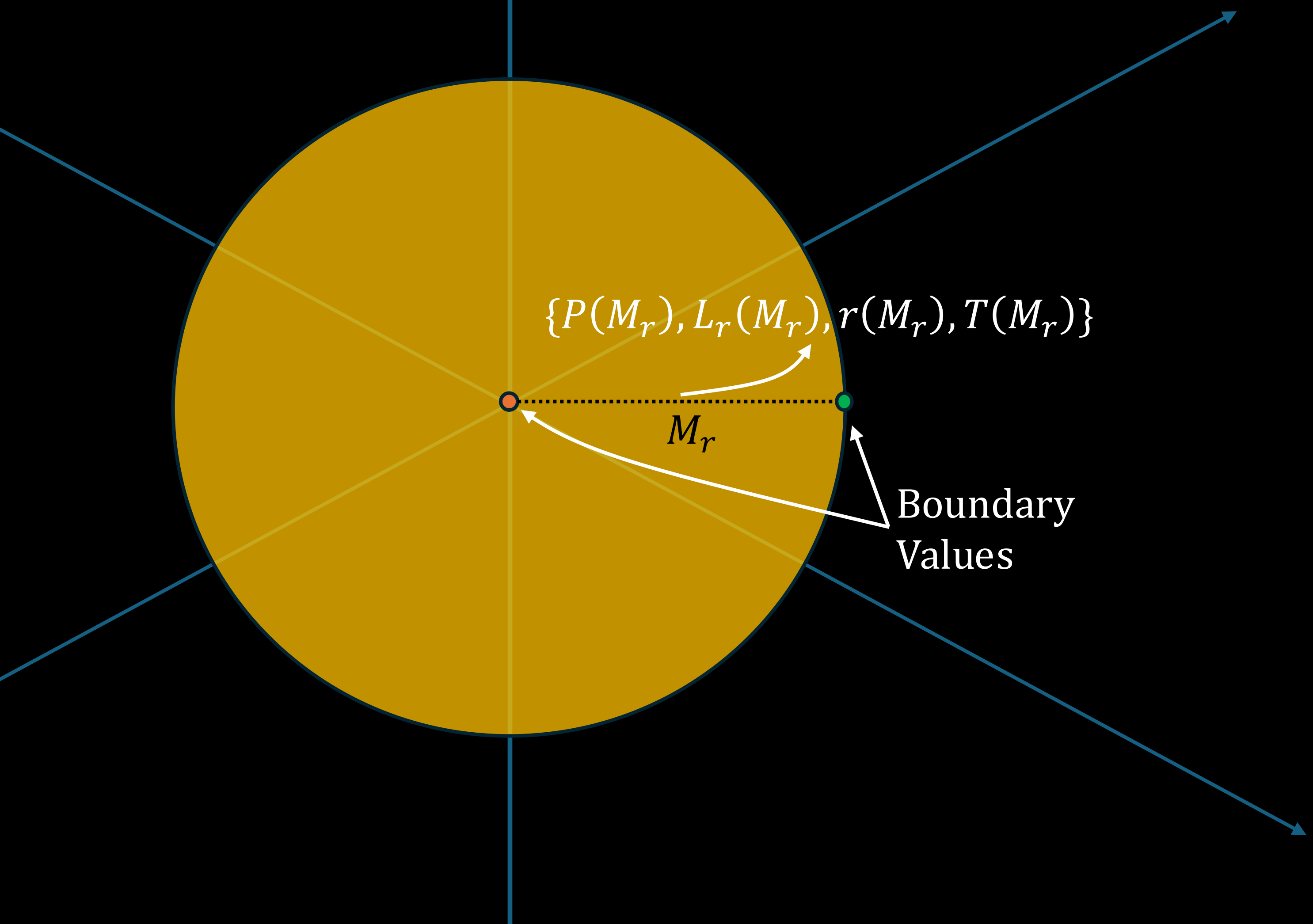
Energy - $\epsilon(M_r)$

Problem Definition





Problem Definition

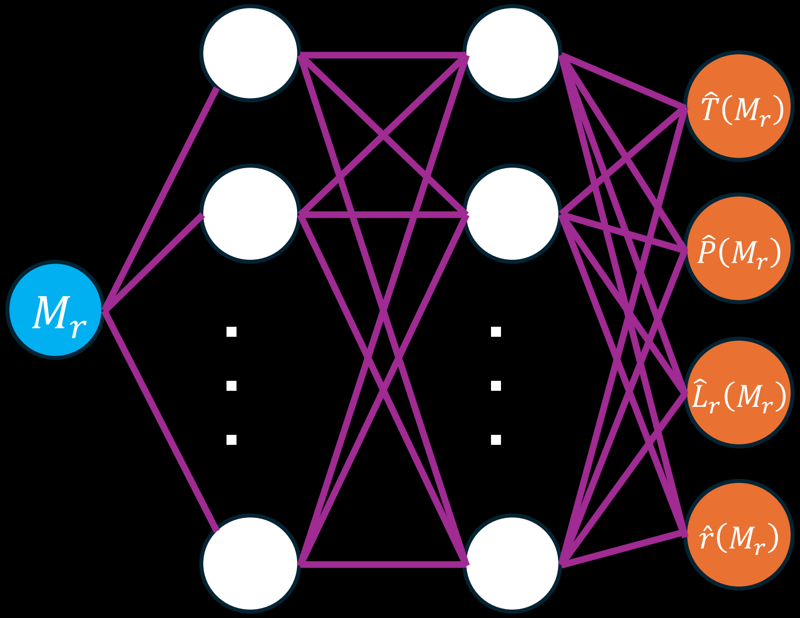


MESA

- Traditionally solved using stellar structure equations and finite difference
- Stellar structure equations are stiff and highly non-linear
- A single simulation can take many hours
- What if we can accelerate this?

(Paxton et al. 2011–2019; Jermyn et al. 2023)

A neural network for stellar structure



Multi-layer Perceptron (MLP)
$f_\theta(M_r)$

Traditionally Trained Using a Loss Function with gradient descent

$$\underbrace{\left(\hat{T}(M_r) - T(M_r)\right)^2 + \left(\hat{P}(M_r) - P(M_r)\right)^2 + \left(\hat{L}_r(M_r) - L_r(M_r)\right)^2 + \left(\hat{r}(M_r) - P(M_r)\right)^2}_{\text{Minimize}}$$

What if we have no data?

Stellar Structure Equations as a Loss Function

$$(1)\ \frac{\partial P}{\partial M_r} = -\frac{\beta_a M_r}{r^4} \qquad \mathcal{R}_P = \frac{\partial \hat{P}}{\partial M_r} + \frac{\beta_a M_r}{\hat{r}^4}$$

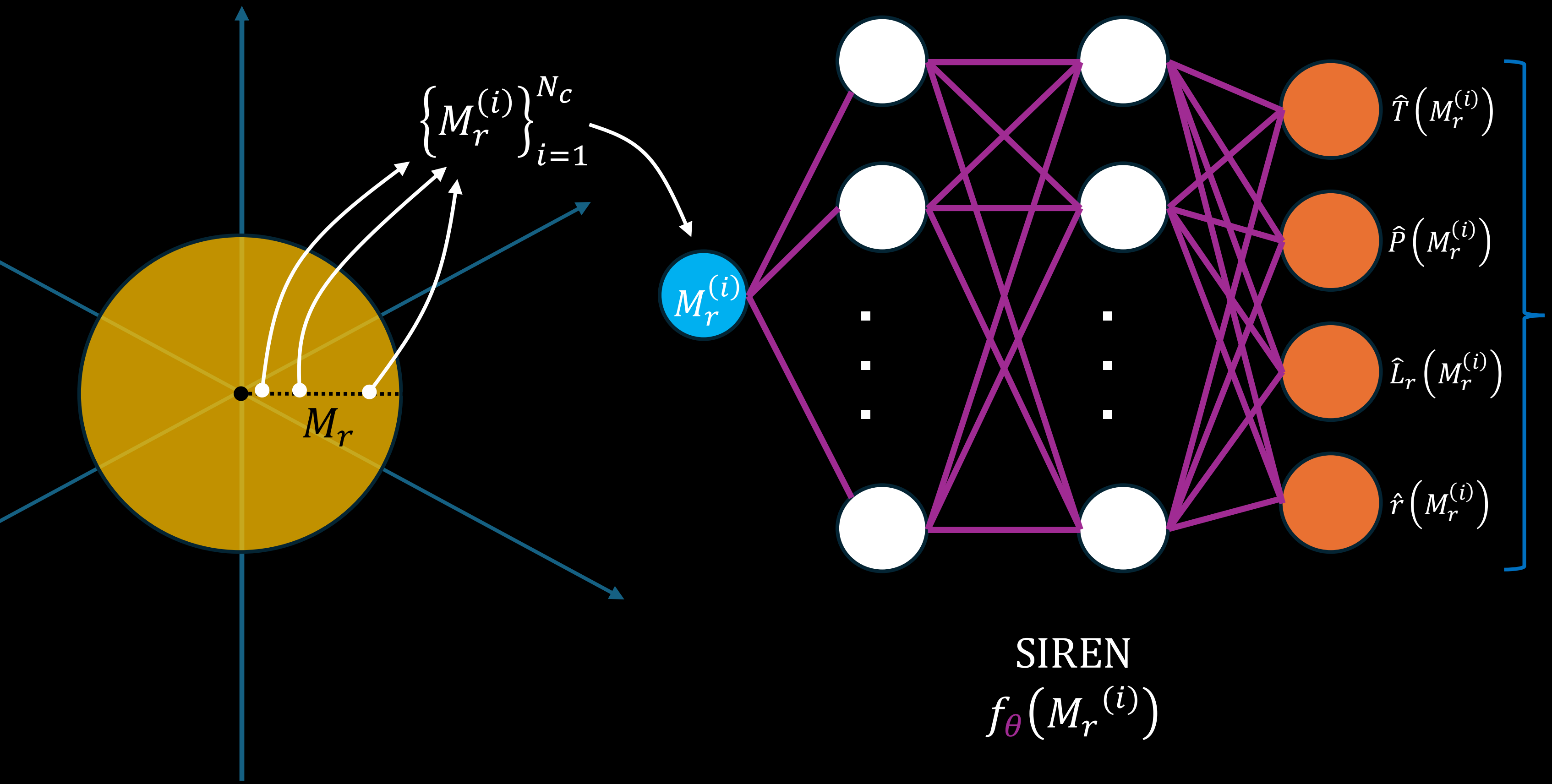
$$(2)\ \frac{\partial r}{\partial M_r} = \frac{\beta_b}{r^2 \rho} \qquad \rightarrow \qquad \mathcal{R}_r = \frac{\partial \hat{r}}{\partial M_r} - \frac{\beta_b}{\hat{r}^2 \rho}$$

$$(3)\ \frac{\partial T}{\partial M_r} = -\frac{\beta_c L_r}{r^4} \nabla(M_r, \kappa) \qquad \mathcal{R}_T = \frac{\partial \hat{T}}{\partial M_r} + \frac{\beta_c \hat{L}_r}{\hat{r}^4} \nabla(M_r, \kappa)$$

$$(4)\ \frac{\partial L_r}{\partial M_r} = \beta_d \epsilon \qquad \mathcal{R}_{L_r} = \frac{\partial \hat{L}_r}{\partial M_r} - \beta_d \epsilon$$

Quantities to Minimize

PINNs for stellar structure



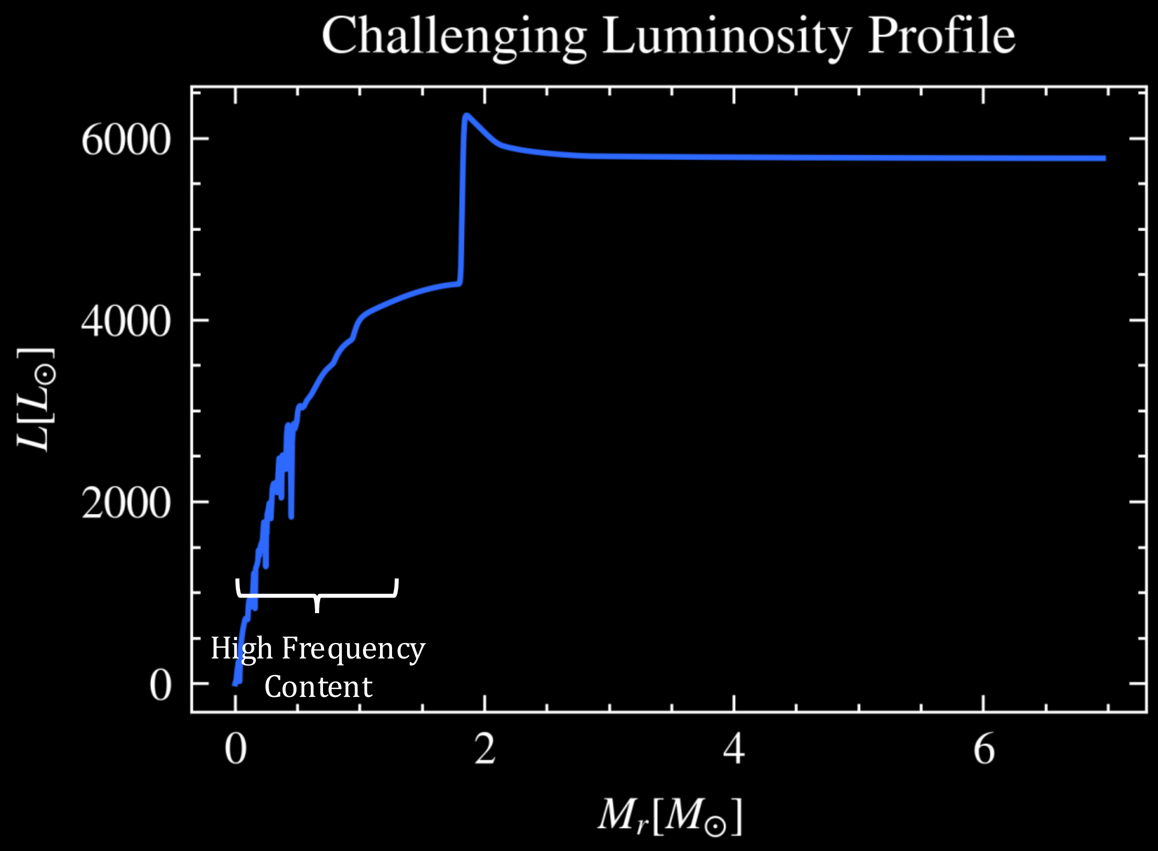
$$\mathcal{L}\left(f_\theta\left(M_r{}^{(i)}\right)\right) = \frac{1}{N_c}\sum_{i=1}^{N_c}\sum_{k\in\{P,r,T,L\}} \alpha_k \left|\mathcal{R}_k\left(M_r{}^{(i)}, f_\theta\left(M_r{}^{(i)}\right)\right)\right|^2$$

V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," arXiv preprint arXiv:2006.09661, 2020.

M. Raissi, P. Perdikaris, and G. 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*, vol. 378, pp. 686–707, 2019.

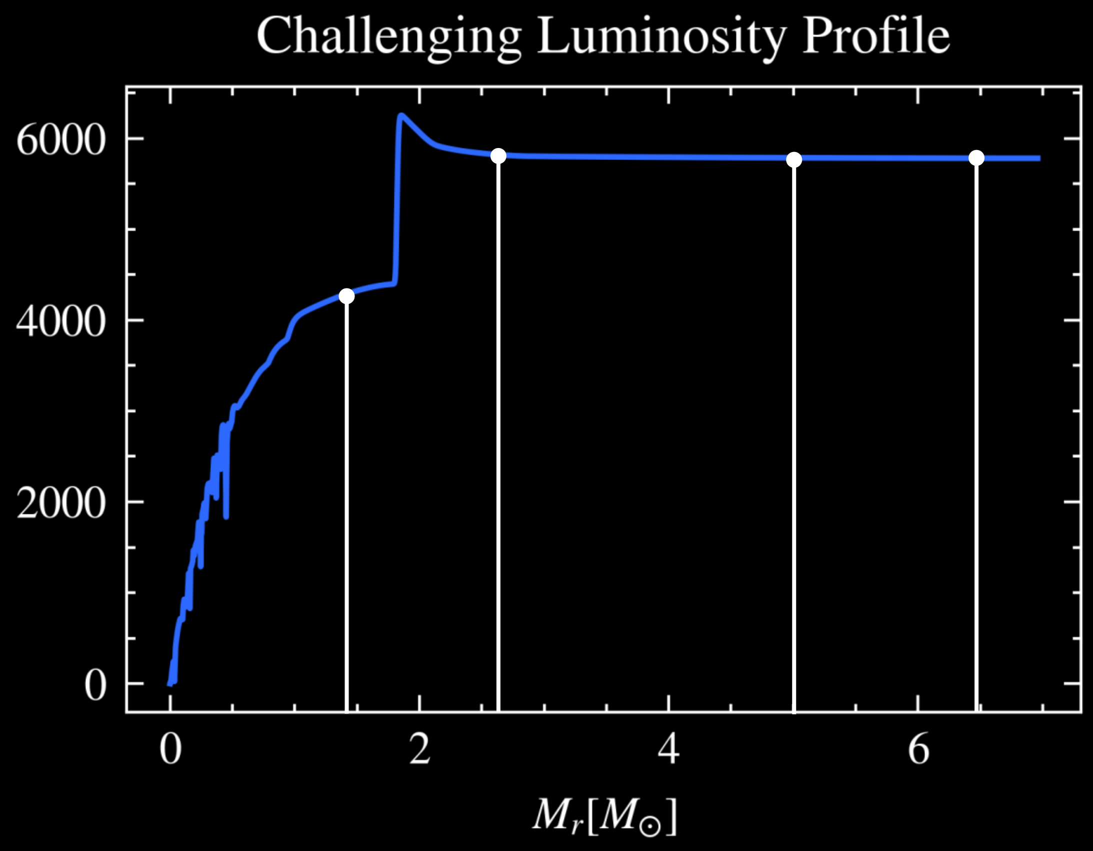
Challenges with PINNs

Spectral bias



A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Curran Associates, Inc., 2007.

Collocation point selection



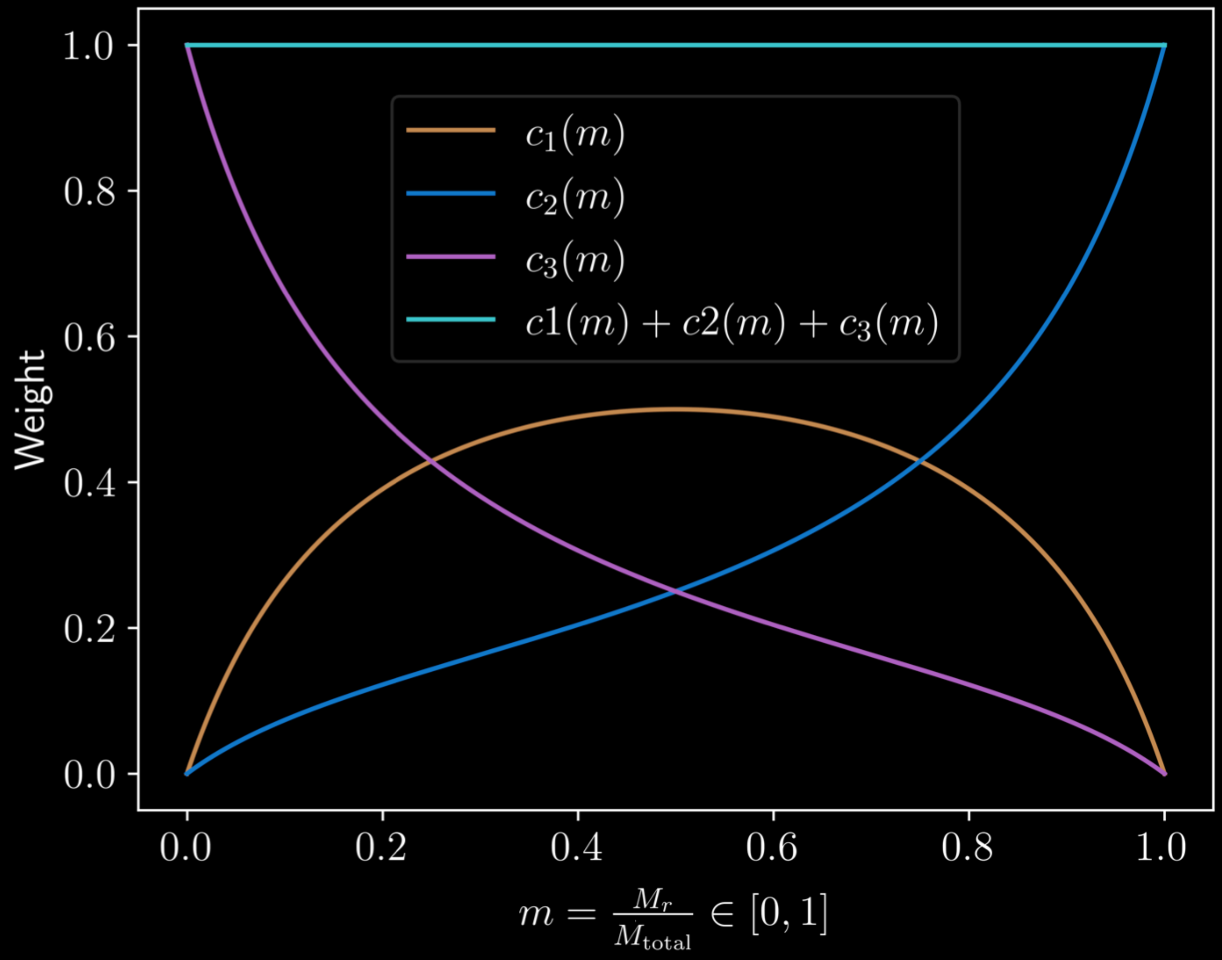
S. J. Anagnostopoulos, J. D. Toscano, N. Stergiopulos, and G. E. Karniadakis, "Residual-based attention and connection to information bottleneck theory in PINNs," arXiv preprint arXiv:2307.00379, 2023.

Boundary Conditions

- $f_\theta\left({M_r}^{(i)}\right)$ is unconstrained and can give values outside of known boundary conditions
- We constraint the output of each variable using the following function:

$$g_\theta(m) = c_1(m) f_\theta(M_r)$$
$$+\, c_2(m)\mathrm{BC}(M_r) + c_3(m)\mathrm{BC}(0),\ m = \frac{M_r}{M_{total}}$$

- This enforces a hard constraint to the output of the system at the boundaries



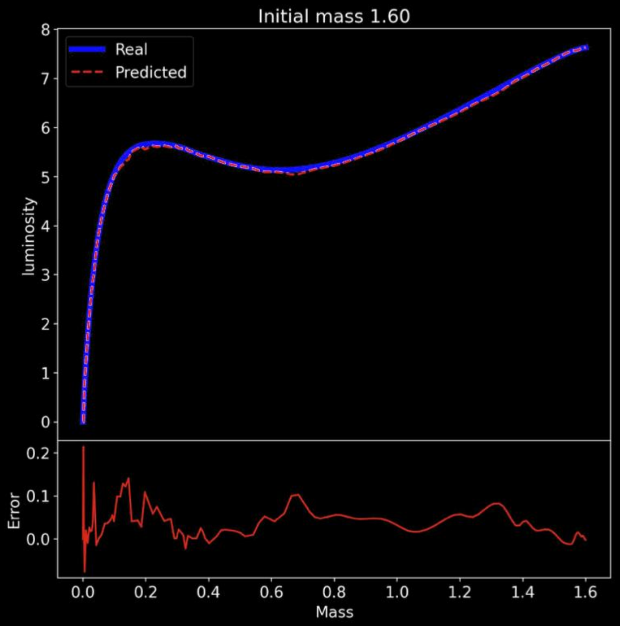
Results

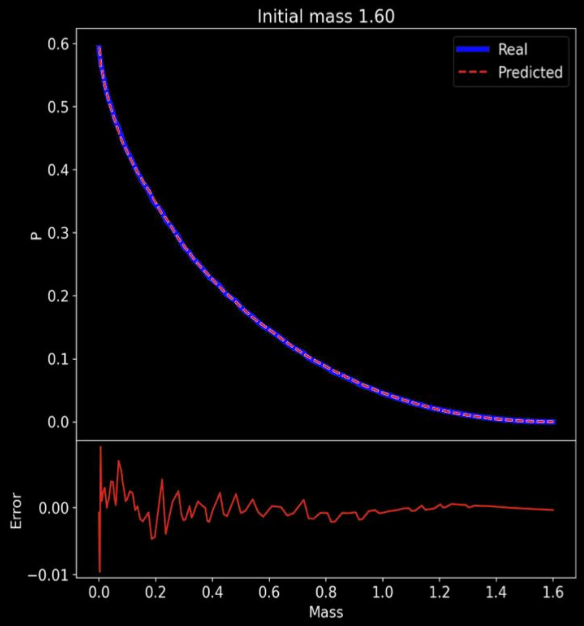
Tested over a range of $M_T = 0.4M_{\odot}$ to $M_T = 9.9M_{\odot}$

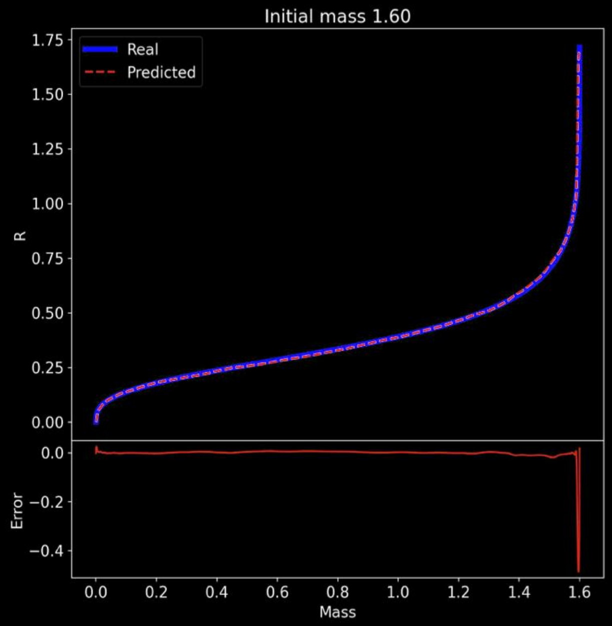
$R^2 = 99.98\%$

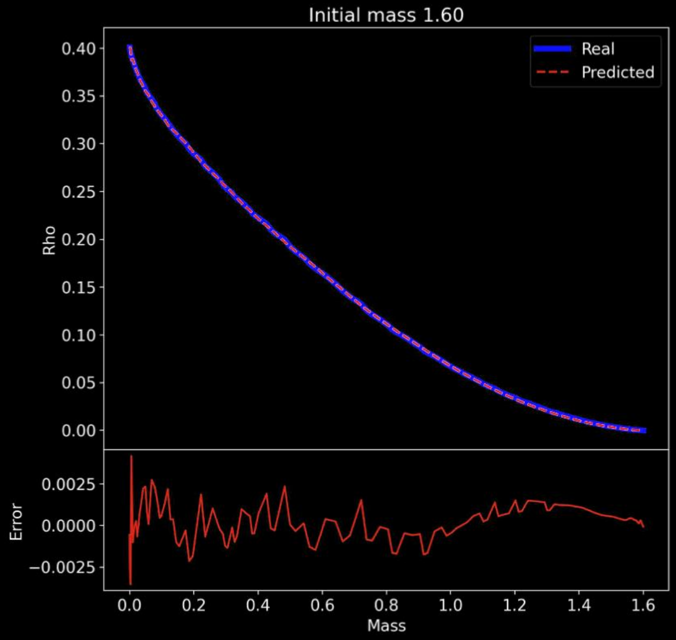
Average MRAE of 3.06%

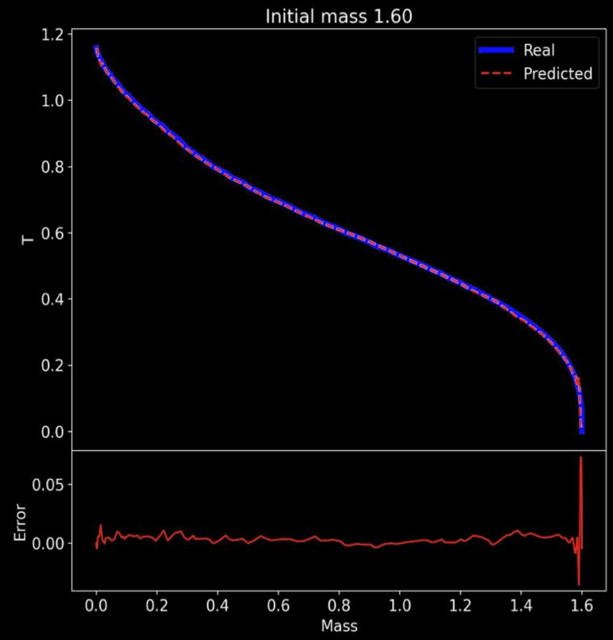
$$MRAE(Q, \hat{Q}) = \frac{1}{N}\sum_{I=1}^{N} \frac{|Q_i - \hat{Q}_i|}{\sigma(Q_i)}$$



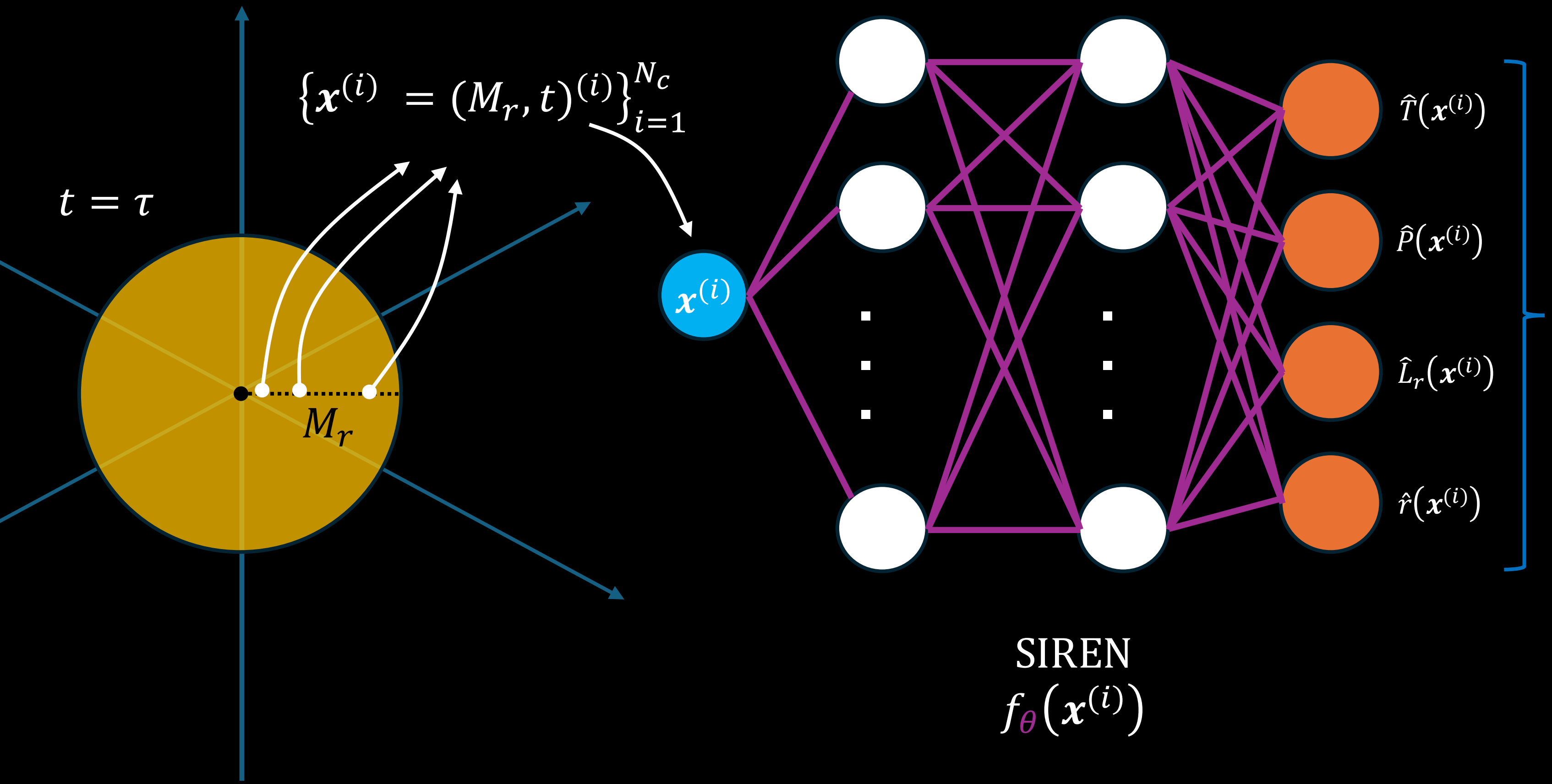








PINNs for stellar structure evolution

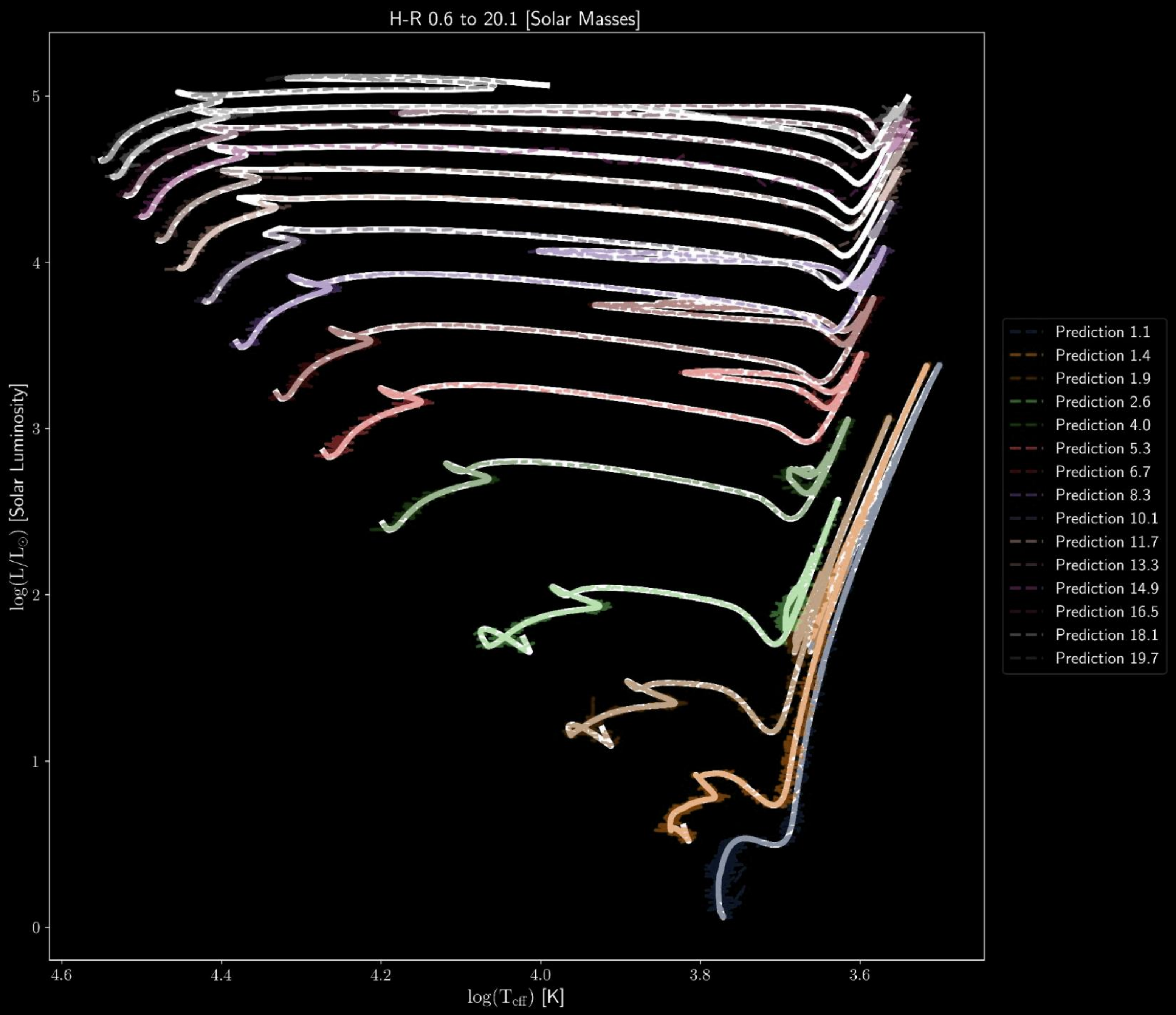


$$\mathcal{L}\left(f_\theta\left(\boldsymbol{x}^{(i)}\right)\right) = \frac{1}{N_c}\sum_{i=1}^{N_c}\sum_{k\in\{P,r,T,L\}} \alpha_k \left|\mathcal{R}_k\left(\boldsymbol{x}^{(i)}, f_\theta\left(\boldsymbol{x}^{(i)}\right)\right)\right|^2$$

M. Ballester *et al.*, "Learning the stellar structure equations via self-supervised physics-informed neural networks," *arXiv preprint arXiv:2604.06255*, 2026.

Results

- We utilize key advancements in the scientific machine learning literature enabling a PINN to be used as a solver for stiff, non-linear equations
- This paves the way to generalizing this system to be able to consider many boundaries with one system
- This work builds the foundation for accelerating stellar structure evolution simulation for both single and binary stars



Ongoing and Future Work

Stellar Structure

- Prediction of ϵ using an auxiliary neural network, rendering the presented PINN "MESA-Free"
- Physics Informed Neural Spline Architecture (PINSA) – Total Mass is given as additional input and generalizes to stellar structures within a trained mass range

Stellar Structure Evolution

- A hybrid approach which solves for stellar structure using a physics informed neural operator (PINO), but utilizes finite difference for computing change in composition
- Neural operator inspired methods for quantity predictions of binary star systems

Looking Forward to Your Questions!

Philipp M. Srivastava

Manuel Ballester

Santiago Lopez-Tapia

Seth Gossage

Patrick Koller

Ugur Demir

Aggelos Katsaggelos

Vicky Kalogera

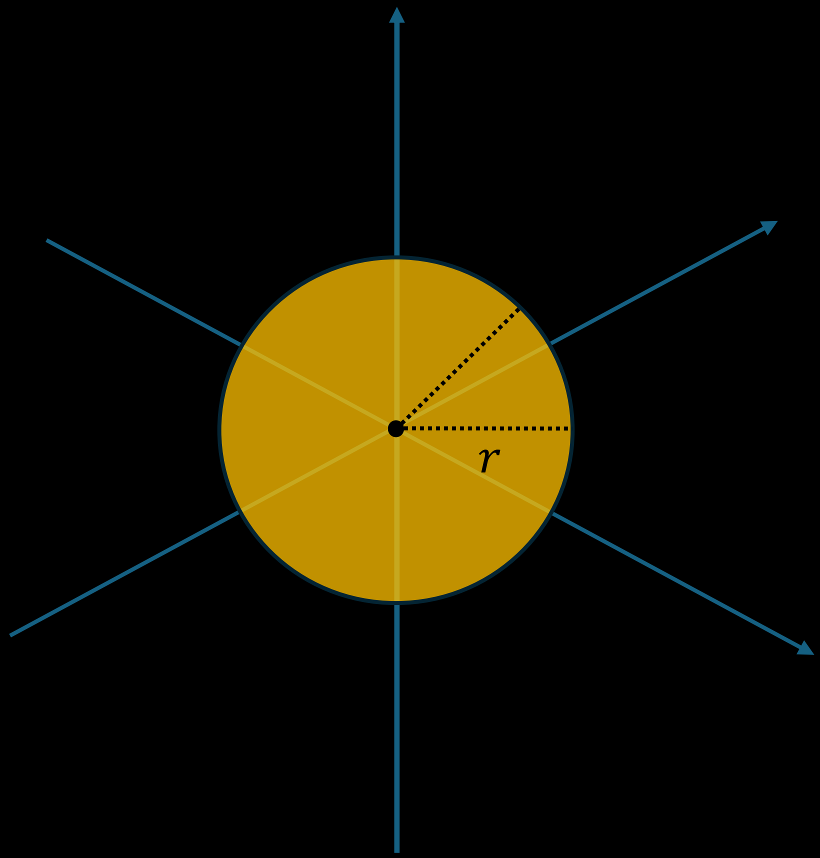
arXiv Paper

Backup Slides

Implementation Details

- Auxiliary networks utilize Random Fourier Features [] and sinusoidal activations []
- PINN uses only sinusoidal activations
- Trained for 10,000 with a batch size of 256 with AdaM [] optimizer with a scaling factor for $\lambda = 5 \times 10^{-2}$
- SP-PINN [] is used to accelerate gradient computation
- Mass range is $M_T = 0.4 M_\odot$ to $M_T = 9.9 M_\odot$ using 96 uniformly spaced samples per profile

Defining Stellar Structure



- Consider temperature varying from the star's center to its surface
- We assume that temperature varying from the center to any point on the surface is the same (spherical symmetry)
- Consider other quantities of interest

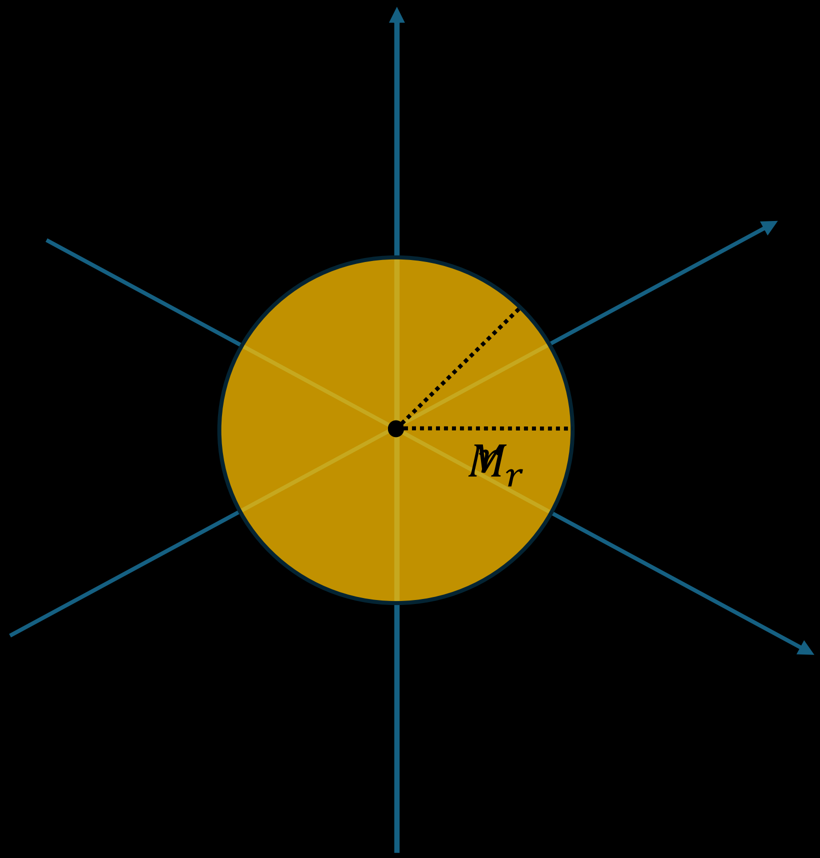
<u>Stellar Structure Quantities</u>

Pressure - $P(r)$
Luminosity - $L_r(r)$
Mass – $M_r(r)$
Temperature – $T(r)$

<u>Auxiliary Quantities</u>

Density – $\rho(r)$
Opacity - $\kappa(r)$
Energy - $\epsilon(r)$

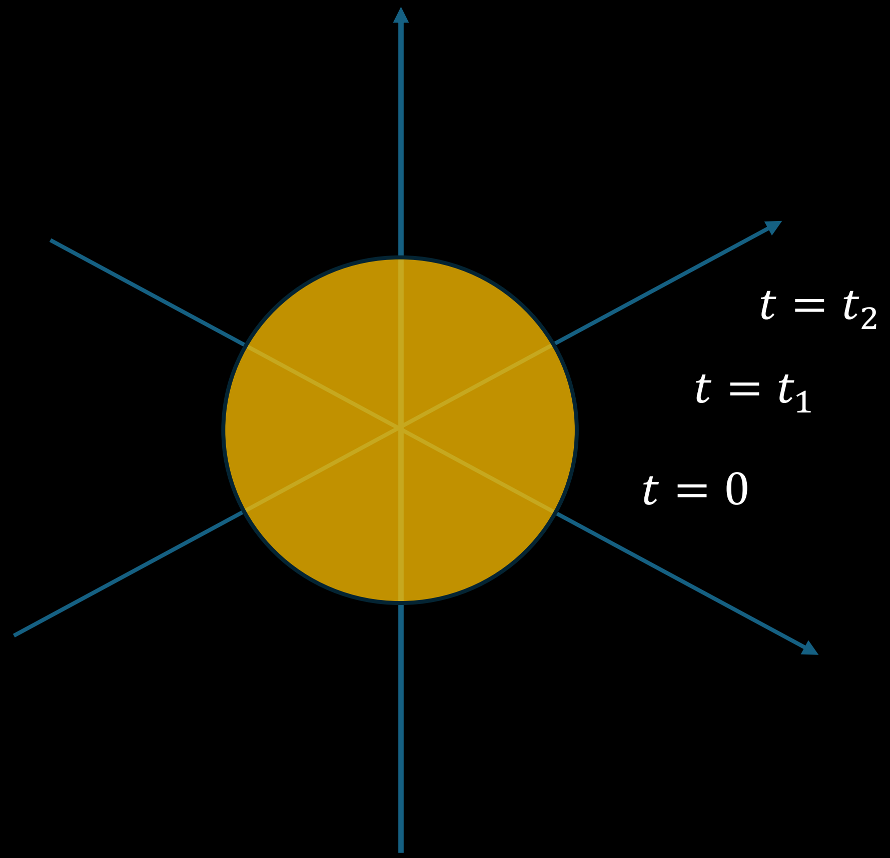
Change of Independent Variables



- Instead of using radius r as a spatial coordinate, we can consider the mass enclosed at r as the spatial coordinate
- We denote the mass enclosed at a given radius as M_r
- This can be done because M_r is monotonically increasing with r

<u>Stellar Structure Quantities</u>	<u>Auxiliary Quantities</u>
Pressure – $P(M_r)$	Density – $\rho(M_r)$
Luminosity – $L_r(M_r)$	Opacity - $\kappa(M_r)$
Radius – $r(M_r)$	Energy - $\epsilon(M_r)$
Temperature – $T(M_r)$	

Stellar Structure Evolution



<u>Stellar Structure Quantities</u>
Pressure – $P(M_r, t)$
Luminosity – $L_r(M_r, t)$
Radius – $r(M_r, t)$,
Temperature – $T(M_r, t)$

<u>Auxiliary Quantities</u>
Density – $\rho(M_r, t)$
Opacity - $\kappa(M_r, t)$
Energy - $\epsilon(M_r, t)$

Stellar Structure
(in Hydrostatic and Thermal Equilibrium)

$$(1)\ \frac{\partial P}{\partial M_r} = \frac{\beta_a M_r}{r^4}$$

$$(2)\ \frac{\partial r}{\partial M_r} = \frac{\beta_b}{r^2 \rho}$$

$$(3)\ \frac{\partial T}{\partial M_r} = -\frac{\beta_c L_r}{r^4} \nabla(M_r, \kappa)$$

$$(4)\ \frac{\partial L_r}{\partial M_r} = \beta_d \epsilon$$

Where,

$$\nabla_{ad} = \frac{d \ln T}{d \ln P}$$

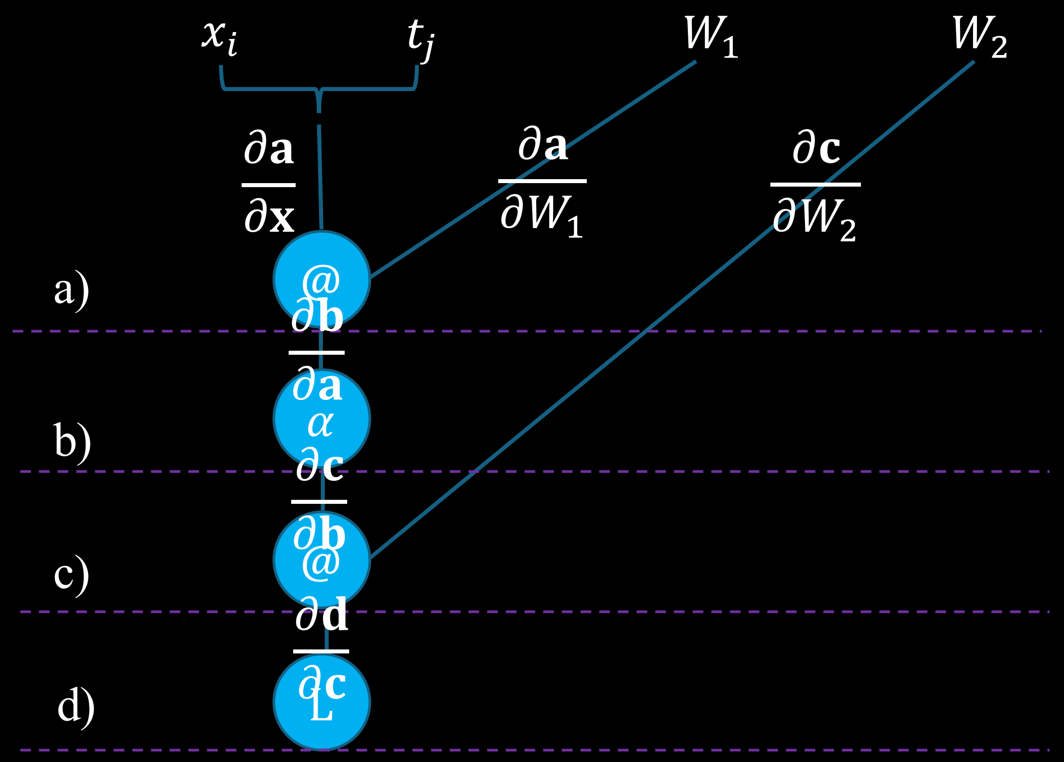
$$\nabla_{rad} = \frac{3\kappa P L_r}{16\pi a c G M_r T^4}$$

and

$$\nabla(M_r, \kappa) = \begin{cases} \nabla_{rad}, \text{if } \nabla_{rad} \leq \nabla_{ad} \\ \nabla_{conv}, \text{if } \nabla_{rad} > \nabla_{ad} \end{cases}$$

How can we enforce PDE residual?

Let $f_\theta(\cdot)$ be a two-layer neural network defined as $f_\theta(\mathbf{x}) = \alpha(\mathbf{x}W_1)W_2$ where $\theta = \{W_1, W_2\}$, $W_1 \in \mathbb{R}^{(2\times d)}$, and $W_2 \in \mathbb{R}^{(d\times 1)}$



- Computational graph is built to compute gradients by default in many libraries
- This makes computing gradients with respect to the input easy
- However, it can be expensive, especially for higher-order derivatives
- Once computed we can enforce it easily in the loss function