Dual-Balancing for Physics-Informed Neural Networks Supplementary Material

Chenhong Zhou[1] , **Jie Chen**[1] , **Zaifeng Yang**[2] and **Ching Eng Png**[2]
[1]Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR, China
[2]Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR), Singapore
{cschzhou, chenjie}@comp.hkbu.edu.hk, {yang_zaifeng, pngce}@ihpc.a-star.edu.sg

A Experiments on More PDEs

A.1 PDE Benchmarks

To further validate the effectiveness of our proposed method, we conduct experiments on solving more PDEs, including Allen-Cahn Equation, the inverse problem for Burgers Equation, and the steady-state Navier-Stokes Equation.

Allen-Cahn Equation

Allen-Cahn PDE is a reaction-diffusion equation that typically arises in phase-field models to describe the evolution of the phase separation process. Allen-Cahn PDE is a challenging benchmark as its solutions have sharp space and time transitions (referred to as stiff PDEs), which has attracted broad interest in the realm of solving PDEs by PINNs [Zhao, 2020; Xiang *et al.*, 2022; McClenny and Braga-Neto, 2023]. Here the Allen-Cahn equation is formulated as follows:

$$u_t - 0.0001u_{xx} + 5u^3 - 5u = 0,\ x \in [-1,1], t \in [0,1], \tag{1}$$

$$u(x,0) = x^2 cos(\pi x), \quad x \in [-1,1], \tag{2}$$

$$u(-1,t) = u(1,t), \quad t \in [0,1], \tag{3}$$

$$u_x(-1,t) = u_x(1,t), \quad t \in [0,1]. \tag{4}$$

Equations (3) and (4) show periodic boundary conditions. We use the exact solution provided in [McClenny and Braga-Neto, 2023] to evaluate the approximation performance of different weighting methods in PINNs.

Inverse Problem for Burgers Equation

Burgers equation is a fundamental PDE in various fields, including nonlinear acoustics, gas dynamics, and fluid mechanics [Raissi *et al.*, 2019]. Here we consider solving the inverse problem in the Burgers equation. The Burgers equation with Dirichlet boundary conditions is described as follows:

$$u_t + C_1 u u_x - C_2 u_{xx} = 0, \quad x \in [-1,1],\ t \in [0,1], \tag{5}$$

$$u(x,0) = -sin(\pi x), \quad x \in [-1,1] \tag{6}$$

$$u(-1,t) = u(1,t) = 0, \quad t \in [0,1] \tag{7}$$

where C_1 and C_2 are the unknown parameters to be identified. To solve the inverse problem, extra observations are needed and produced by randomly selecting from the exact solutions provided in [Raissi *et al.*, 2019]. The underlying true parameters are $C_1 = 1$ and $C_2 = 0.01/\pi$. We aim to train a PINN for both accurate approximation of $u(x,t)$ and parameter identification of C_1 and C_2.

Navier-Stokes Equation

The lid-driven cavity flow problem governed by the incompressible Navier-Stokes (NS) equations is a classic benchmark in the field of computational fluid dynamics [Wang *et al.*, 2021]. Here we consider the steady-state flow in a two-dimensional lid-driven cavity, which can be formulated as below:

$$\boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \frac{1}{Re}\Delta \boldsymbol{u} = 0, \qquad \boldsymbol{x} \text{ in } \Omega, \tag{8}$$

$$\nabla \cdot \boldsymbol{u} = 0, \qquad \boldsymbol{x} \text{ in } \Omega, \tag{9}$$

$$\boldsymbol{u}(\boldsymbol{x}) = (1,0), \qquad \boldsymbol{x} \text{ on } \Gamma_1, \tag{10}$$

$$\boldsymbol{u}(\boldsymbol{x}) = (0,0), \qquad \boldsymbol{x} \text{ on } \Gamma_0, \tag{11}$$

where $\boldsymbol{x} = (x,y) \in \Omega = [0,1] \times [0,1]$, and $\boldsymbol{u}(\boldsymbol{x}) = (u(\boldsymbol{x}), v(\boldsymbol{x}))$ is a velocity vector field. $u(\boldsymbol{x})$ and $v(\boldsymbol{x})$ represent the x and y component of the velocity field. $p(\boldsymbol{x})$ is a scalar pressure field, and the Reynolds number is 100. In addition, Γ_1 is the top boundary and Γ_0 denotes all other boundaries. Following [Wang *et al.*, 2021], we aim to train a PINN to predict a latent scalar potential function $\psi(x,y)$ and the pressure field $p(x,y)$. Specifically, the incompressible velocity field can be represented as $(u,v) = (\partial\psi/\partial y, -\partial\psi/\partial x)$. The reference solutions of (u,v) are provided in [Wang *et al.*, 2021] by finite difference methods. Given the distinct boundary conditions on Γ_0 and Γ_1, the BC loss is decomposed into two parts in the DB-PINN, so that inner-balancing can be carried out to derive more fine-grained weights based on the fitting progress of different boundaries.

A.2 Experimental Setting

Experimental setups for all six PDE benchmarks are reported in Table 1. Observational data are needed for solving the inverse problem for Burgers equation in PINNs, and thus we introduce $N^d = 300$ observations for computing the data loss $\mathcal{L}^d$. Moreover, we additionally use L-BFGS optimizers [Liu and Nocedal, 1989] after training 5k iterations of Adam for this inverse problem.

A.3 Experimental Results

Allen-Cahn Equation

The results for solving Allen-Cahn PDE are presented in Table 2. Equal weighting fails to yield accurate predictive solutions, with a relative L^2 error exceeding 50% and a mean

PDE	Network		Dataset				Training strategy	
	Depth	Width	N^f	N^{bc}	N^{ic}	N^d	Optimizer-Epochs	LR-decay
Klein-Gordon	3	50	10000	2000	1000	–	Adam-20k	0.1 per 10k epochs
Wave	5	500	20000	2500	2500	–	Adam-80k	0.9 per 2k epochs
Helmholtz	4	50	20000	400	–	–	Adam-30k	0.1 per 10k epochs
Allen-Cahn	4	128	20000	200	512	–	Adam-80k	–
Burgers	8	20	10000	200	100	300	Adam-5k + L-BFGS	–
Navier-Stokes	3	50	20000	400	–	–	Adam-40k	–

Table 1: Experimental setups for each benchmark PDE.

Method		Allen-Cahn Equation		Burgers Equation		Navier-Stokes Equation	
		L2RE (%)	MAE (%)	L2RE (%)	MAE (%)	L2RE (%)	MAE (%)
Equal weighting		52.439 ± 9.180	16.474 ± 3.820	1.651 ± 0.588	0.240 ± 0.055	10.661 ± 1.382	1.893 ± 0.345
Uncertainty weighting		55.755 ± 8.406	17.727 ± 3.396	2.006 ± 1.132	0.303 ± 0.103	14.869 ± 5.183	2.654 ± 0.953
SA-PINN		5.815 ± 1.311	**1.065 ± 0.279**	0.775 ± 0.516	0.163 ± 0.070	8.183 ± 1.479	1.362 ± 0.388
GW-PINN	mean	10.968 ± 4.945	2.996 ± 1.499	0.812 ± 0.738	0.112 ± 0.035	5.597 ± 0.964	0.982 ± 0.262
	std	15.081 ± 12.831	3.960 ± 3.708	2.702 ± 1.700	0.403 ± 0.292	6.688 ± 1.900	1.130 ± 0.452
	kurtosis	53.039 ± 8.753	16.748 ± 3.420	1.830 ± 1.364	0.293 ± 0.270	6.048 ± 1.744	0.992 ± 0.402
DB-PINN	mean	6.788 ± 1.422	1.741 ± 0.574	0.297 ± 0.257	0.062 ± 0.017	**5.034 ± 0.554**	**0.787 ± 0.196**
	std	**5.510 ± 0.962**	1.200 ± 0.230	**0.253 ± 0.090**	**0.058 ± 0.013**	5.561 ± 0.726	0.869 ± 0.215
	kurtosis	6.809 ± 1.231	1.603 ± 0.352	0.461 ± 0.351	0.074 ± 0.027	5.459 ± 0.600	0.924 ± 0.198

Table 2: Errors (mean ± std) of different weighting methods for solving these three PDE benchmarks.

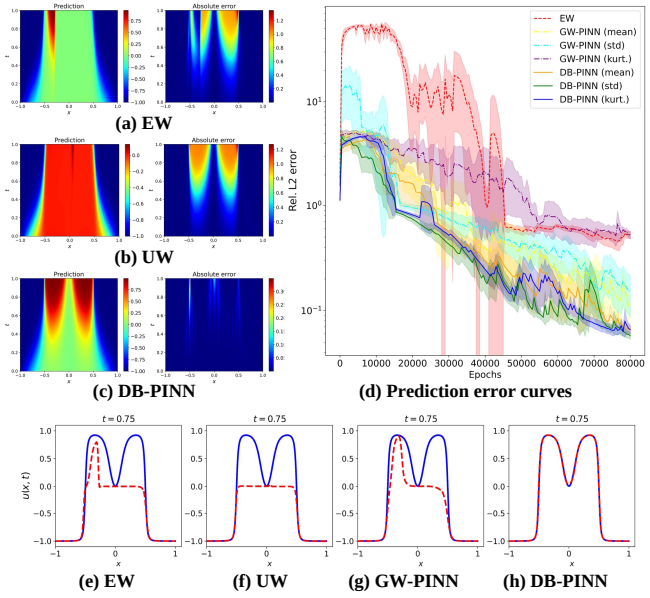

Figure 1: The Allen-Cahn equation. (a-c) The predictions and errors. (d) Prediction error curves. (e-h) Comparison between reference solutions (blue) and predictions (red).

absolute error surpassing 15%. Uncertainty weighting yields worse performance than the baseline, while SA-PINN obtains a prominent boost in performance. Note that GW-PINN achieves a noticeable accuracy boost when using mean and std metrics, but a poorer accuracy when using kurtosis, which shows the importance of choosing the appropriate statistics. In contrast, DB-PINN exhibits outstanding performance across all three statistical measures. Importantly, DB-PINN with std achieves a comparable performance with the lowest L^2 error of 5.5%. Figure 1(a-c) shows the predictions and errors of EW, UW, and DB-PINN (kurtosis). DB-PINN yields the most accurate predictions with the lowest maximum errors. Figure 1(e-h) shows that DB-PINN (kurtosis) exhibits the most accurate prediction, approaching the reference solutions closely. Figure 1(d) displays the prediction error curves of different weighting methods during training. Compared with other methods, DB-PINN consistently has a faster convergence speed and stable predictive performance with low errors regardless of different gradient statistics.

Method		Identified values		Errors ($\times 10^{-3}$)	
		$\hat{C}_1$	$\hat{C}_2$	$\lvert\hat{C}_1 - C_1\rvert$	$\lvert\hat{C}_2 - C_2\rvert$
EW		0.999207	0.004082	0.792	0.898
UW		0.995695	0.004166	4.305	0.983
SA-PINN		0.996673	0.003537	3.327	0.354
GW-PINN	mean	0.999595	0.003488	0.405	0.305
	std	0.997192	0.006269	2.808	3.086
	kurt.	0.989413	0.005575	10.587	2.392
DB-PINN	mean	0.999603	0.003280	0.397	0.097
	std	0.999588	0.003268	0.412	**0.085**
	kurt.	0.999752	0.003364	**0.248**	0.181

Table 3: The identified values ($\hat{C}_1$, $\hat{C}_2$) for unknown parameters (C_1, C_2) in Burgers equation. True values: $C_1 = 1$, $C_2 = 0.01/\pi \approx 0.003183$.

Burgers Equation

Table 2 compares the predictive performance of different weighting methods to solve the Burgers equation. GW-PINN decreases the reconstruction errors using mean while performing poorly when using std and kurtosis. This reveals the

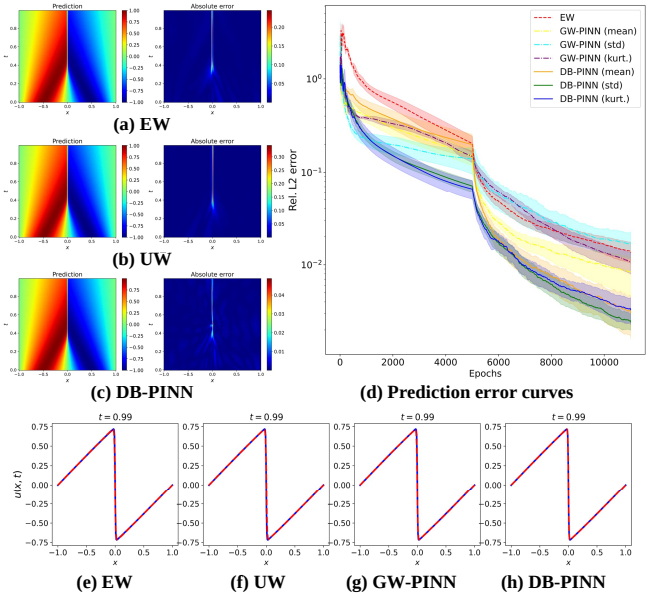

Figure 2: The Burgers equation. (a-c) The predictions and errors. (d) Prediction error curves. (e-h) Comparison between reference solutions (blue) and predictions (red).

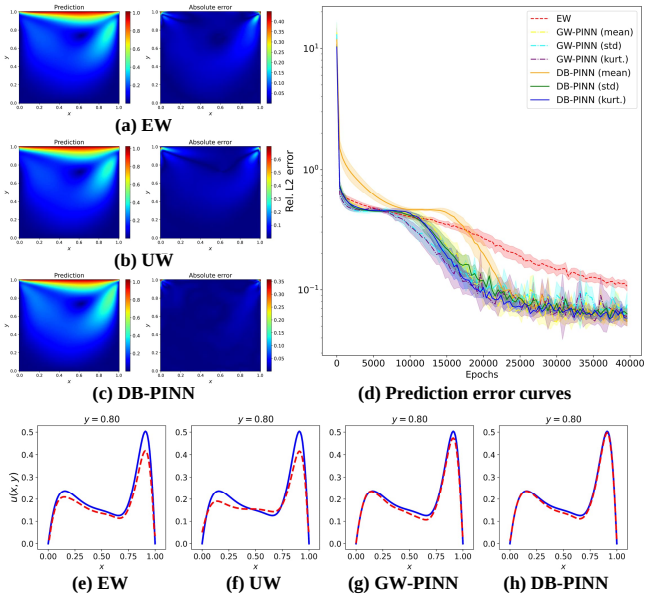

Figure 3: The Navier-Stokes equation. (a-c) The predictions and errors. (d) Prediction error curves. (e-h) Comparison between reference solutions (blue) and predictions (red).

importance of choosing the appropriate gradient statistics in GW-PINN, while the performance of DB-PINN is relatively stable and does not vary significantly with different gradient statistics. When utilizing identical statistical metrics, DB-PINN consistently delivers superior performance than GW-PINN, with a more evident advantage observed in std and kurtosis. Furthermore, we assess the capability of parameter identification and report the results in Table 3. The identified parameter values of DB-PINN exhibit the closest proximity to the true values. Therefore, the efficacy of DB-PINN in approximating the solutions and accurately identifying unknown parameters demonstrates its effectiveness in handling inverse problems. Figure 2(a-c) shows that DB-PINN (kurtosis) achieves the lowest maximum errors. Figure 2(e-h) displays that the corner predictions of DB-PINN (kurtosis) are more accurate than those of other methods. Figure 2(d) shows the prediction error curves along with training epochs. Importantly, all the errors rapidly drop after 5000 iterations of Adam, and DB-PINN exhibits the fastest convergence and the least predictive errors.

Navier-Stokes Equation

Table 2 reports the results of the Navier-Stokes equation. GW-PINN using any one of the statistical metrics achieves better performance than the learning approach (both UW and SA-PINN). In particular, DB-PINN further exceeds GW-PINN regardless of statistical metrics, which validates its effectiveness in solving time-independent PDEs. Figure 3(a-c) shows that DB-PINN yields the most accurate predictions, which is also proved by comparing the prediction results of snapshots in Figure 3(e-h). Figure 3(d) shows that DB-PINN achieves stable convergence performance with small standard deviations of errors.

B More Results Visualization

B.1 Loss And Weight Curves During Training

We provide the visualization of loss and weight curves for four time-dependent PDEs: Klein-Gordon Equation, Wave Equation, Allen-Cahn Equation, and the inverse problem of Burgers Equation; and two time-independent PDEs: Helmholtz Equation and steady-state Navier-Stokes Equation in the below figures. It can be concluded that DB-PINN effectively narrows the gap between BC loss and IC loss, facilitating convergence to relatively similar values across various conditions. This is a clear advantage compared to GW-PINN, as DB-PINN prevents training from continuously favoring the convergence of a certain easy condition and emphasizes the contributions of other under-fitting condition losses according to the current training progress. We observe that GW-PINN using std or kurtosis metrics shows poor performance for solving the Wave equation and Allen-Cahn equation. This is due to abrupt spikes in weight curves, as shown in Figure 5(e,g) and Figure 6(e,g), thereby leading to unstable training and poor performance. Instead, DB-PINN is capable of resisting large variances during the weight update process and achieves smooth weight curves for stable training. Furthermore, Figure 7 displays the identified values along with training epochs for the inverse problem in Burgers equation. It can be found that DB-PINN accelerates the accurate identification of unknown parameters and obtains relatively accurate C_1 and C_2 values during training with Adam from Figure 7(e,f,g,h). In summary, our proposed DB-PINN has convincingly showcased its superiority in enhancing the predictive accuracy and convergence speed of PINNs.

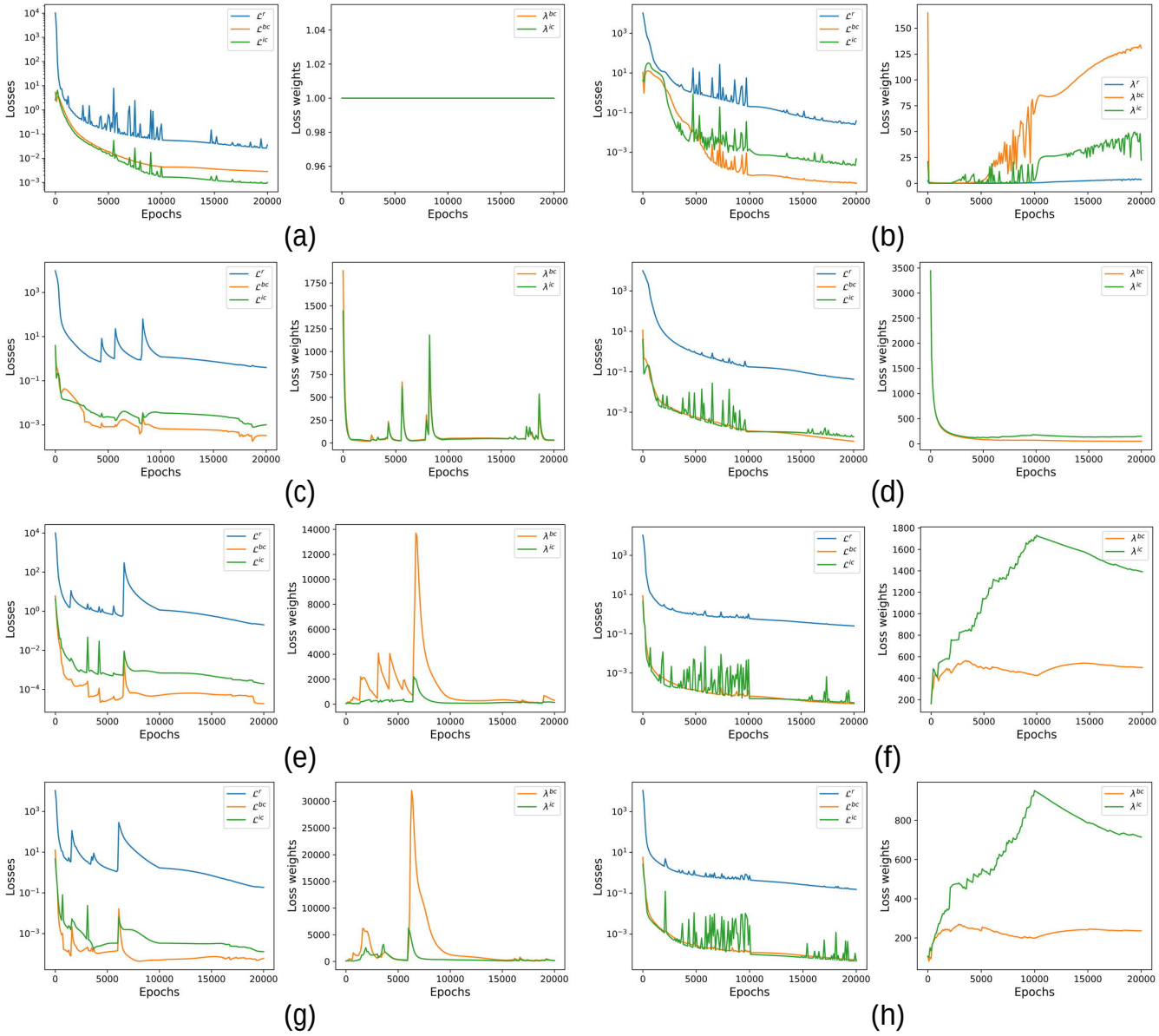

Figure 4: Loss and weight curves of the Klein-Gordon equation: (a) EW, (b) UW, (c) GW-PINN (mean), (d) DB-PINN (mean), (e) GW-PINN (std), (f) DB-PINN (std), (g) GW-PINN (kurtosis), (h) DB-PINN (kurtosis).

References

[Liu and Nocedal, 1989] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[McClenny and Braga-Neto, 2023] Levi D McClenny and Ulisses M Braga-Neto. Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474:111722, 2023.

[Raissi *et al.*, 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.

[Wang *et al.*, 2021] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[Xiang *et al.*, 2022] Zixue Xiang, Wei Peng, Xu Liu, and Wen Yao. Self-adaptive loss balanced physics-informed neural networks. *Neurocomputing*, 496:11–34, 2022.

[Zhao, 2020] Colby L Zhao. Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed

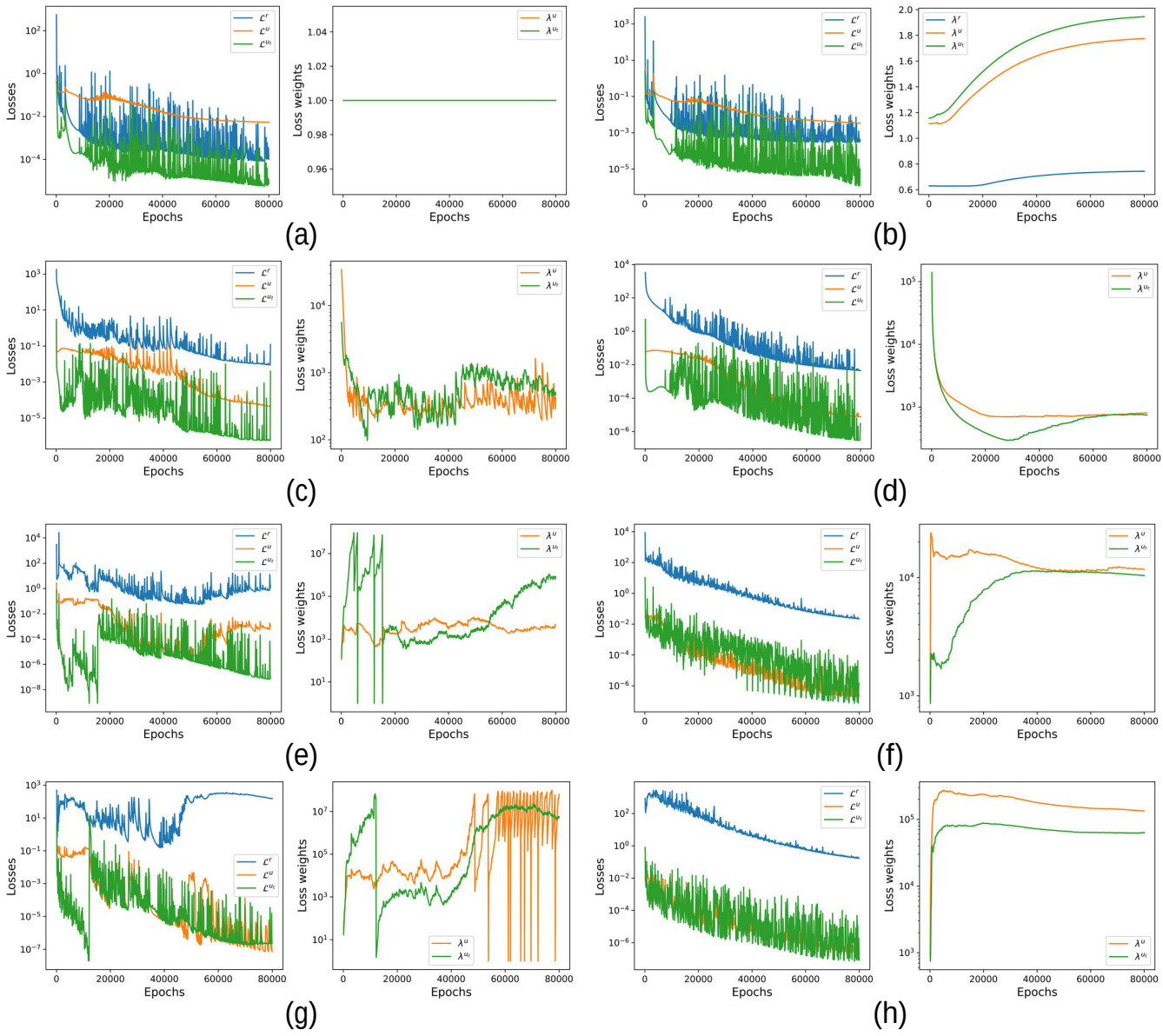

Figure 5: Loss and weight curves of the Wave equation: (a) EW, (b) UW, (c) GW-PINN (mean), (d) DB-PINN (mean), (e) GW-PINN (std), (f) DB-PINN (std), (g) GW-PINN (kurtosis), (h) DB-PINN (kurtosis).

neural networks. *Communications in Computational Physics*, 29(3), 2020.

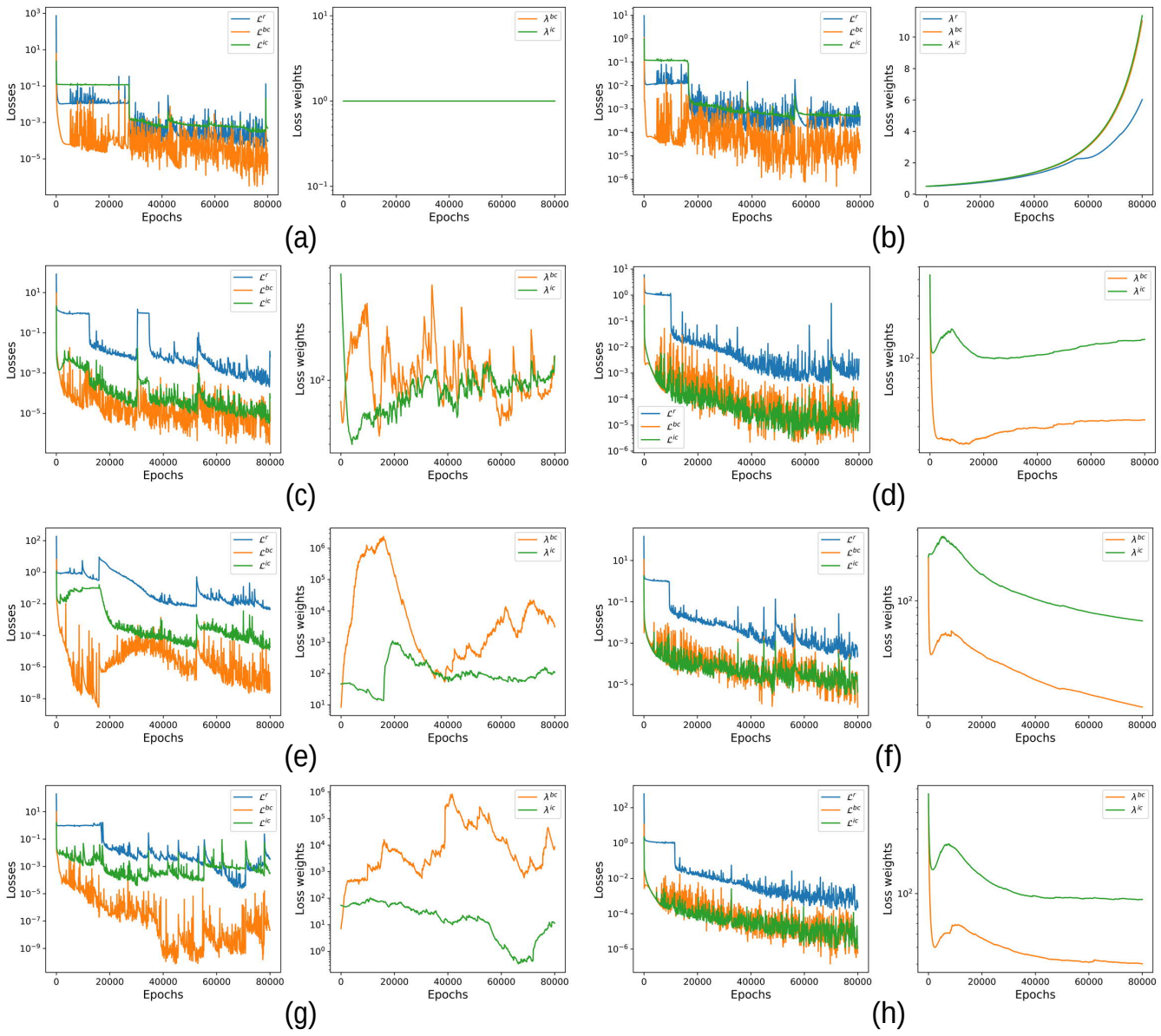

Figure 6: Loss and weight curves of the Allen-Cahn equation: (a) EW, (b) UW, (c) GW-PINN (mean), (d) DB-PINN (mean), (e) GW-PINN (std), (f) DB-PINN (std), (g) GW-PINN (kurtosis), (h) DB-PINN (kurtosis).

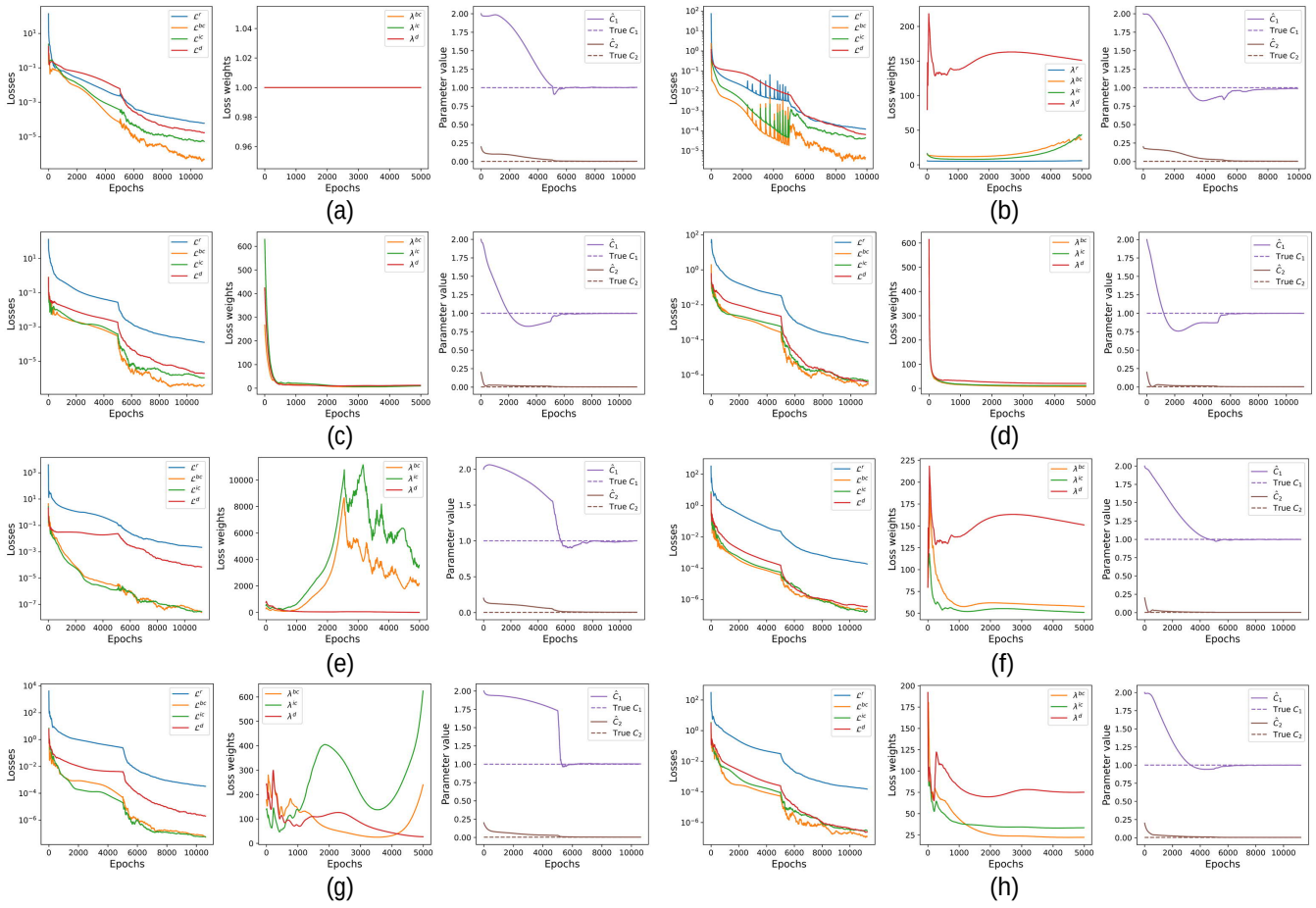

Figure 7: Loss curves, weight curves, and the identified values of unknown constants for the Burgers equation: (a) EW, (b) UW, (c) GW-PINN (mean), (d) DB-PINN (mean), (e) GW-PINN (std), (f) DB-PINN (std), (g) GW-PINN (kurtosis), (h) DB-PINN (kurtosis).

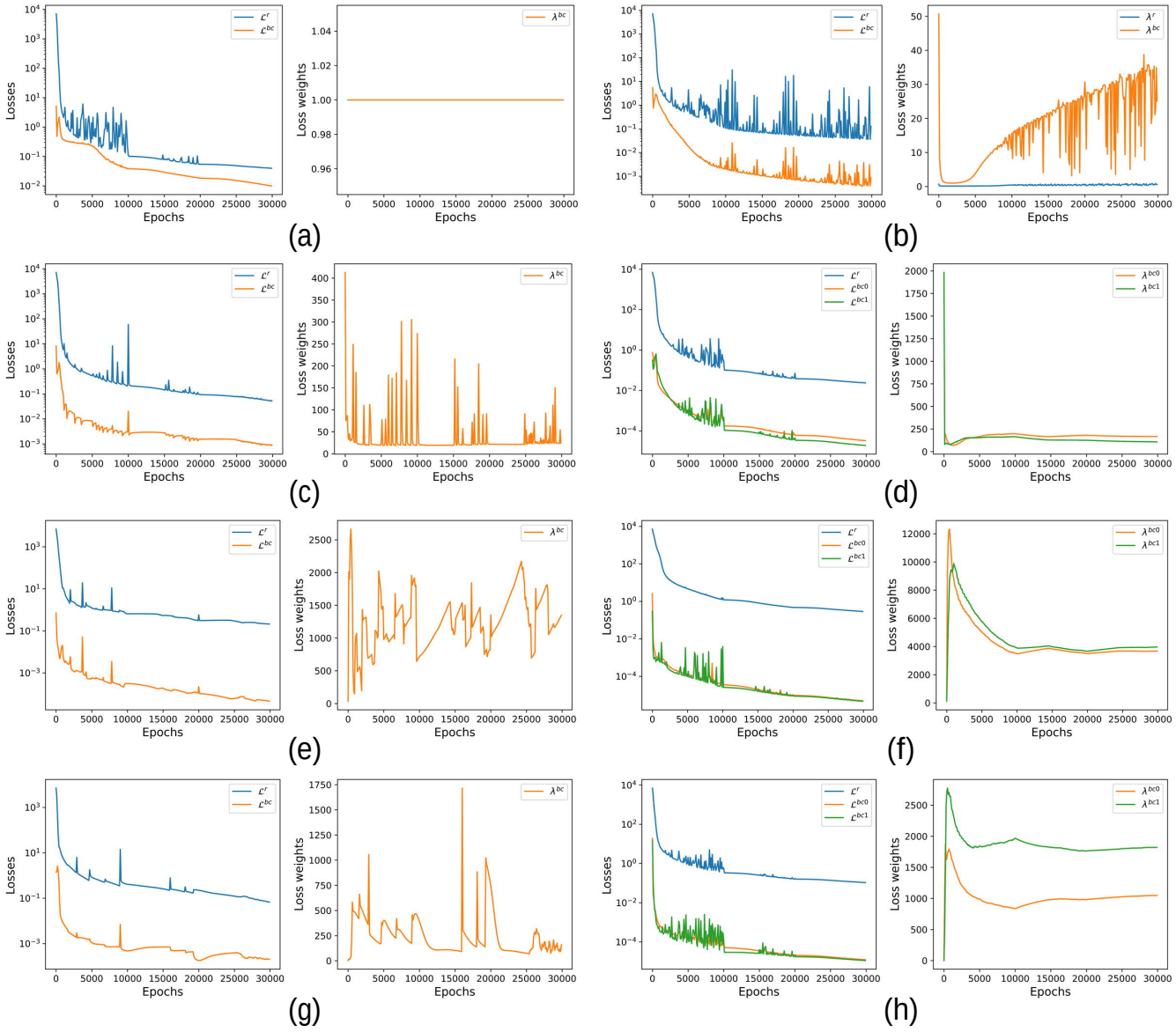

Figure 8: Loss and weight curves of the Helmholtz equation: (a) EW, (b) UW, (c) GW-PINN (mean), (d) DB-PINN (mean), (e) GW-PINN (std), (f) DB-PINN (std), (g) GW-PINN (kurtosis), (h) DB-PINN (kurtosis).

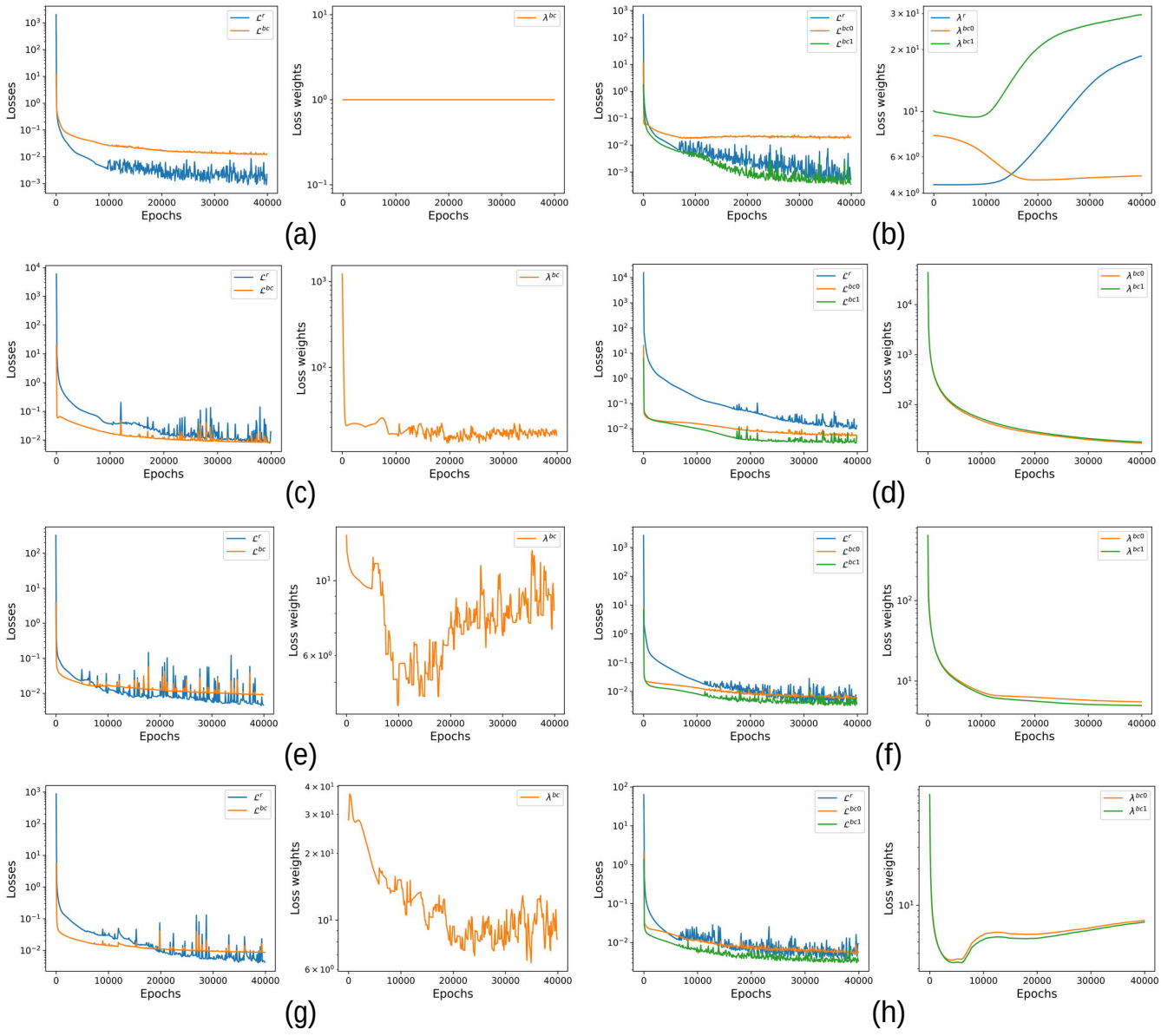

Figure 9: Loss and weight curves of the Navier-Stokes equation: (a) EW, (b) UW, (c) GW-PINN (mean), (d) DB-PINN (mean), (e) GW-PINN (std), (f) DB-PINN (std), (g) GW-PINN (kurtosis), (h) DB-PINN (kurtosis).