Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный университет»

Физический факультет

Кафедра вычислительной физики

Выбор гиперпараметров нейронной сети для решения задач математической физики

Выполнил: Поляков Даниил Николаевич
Научный руководитель: Степанова Маргарита Михайловна

Санкт-Петербург, 2023

Введение

Самыми распространёнными классами методов численного решения дифференциальных уравнений являются метод конечных разностей (МКР) и метод конечных элементов (МКЭ).

Нейронные сети набирают обороты во многих областях науки и техники. В частности, поскольку нейронная сеть является универсальным аппроксиматором, с её помощью можно аппроксимировать функции и решать дифференциальные уравнения (**PINN**).

Для получения наилучшего результата перед обучением модели необходимо правильно подобрать конфигурацию её гиперпараметров. Процесс подбора конфигурации модели часто оказывается очень трудоёмким. В связи с этим в настоящее время ведутся исследования и разработки в области автоматической оптимизации гиперпараметров (*Hyper-Parameter Optimization*, **HPO**).

В настоящей работе исследуются процессы ручной и автоматической оптимизации гиперпараметров нейронной сети для решения уравнения Гельмгольца в пространствах больших размерностей (до 8). Построение и тренировка модели проводится на базе фреймворка **PyTorch**, а автоматическая оптимизация гиперпараметров проводится с использованием фреймворка **Ray Tune**.

Цели работы

Цели, поставленные в текущей работе:

1. изучить влияние гиперпараметров на сходимость нейронной сети к решению;
2. сравнить методы автоматического подбора гиперпараметров;
3. исследовать возможность решения уравнения Гельмгольца в пространстве большой размерности.

Искусственная нейронная сеть

В настоящей работе ведётся работа с искусственными нейронными сетями с прямой связью (*Feed-forward Neural Network*, FNN). Нейронные сети представляют собой композицию линейных и нелинейных преобразований. Элементарным элементом нейронной сети является нейрон. В случае полносвязной нейронной сети значение, возвращаемое нейроном, выражается следующим образом:

$$y_j = \sigma_j\left(b_j + \sum_{i}^{n} w_{ji} x_i\right)$$

x_i — значения на выходе из нейронов предыдущего слоя,
w_{ji} — весовые коэффициенты, b_j — сдвиги, σ_j — функция активации.

Многослойная (глубокая) нейронная сеть выражается суперпозицией подобных преобразований:

$$f(\boldsymbol{x}) = \varphi_L(\ldots\varphi_2(\varphi_1(\boldsymbol{x})))$$

Градиентный спуск:

$$\theta_i = \theta_{i-1} - lr \cdot \nabla_\theta \mathrm{loss}(\theta_{i-1})$$

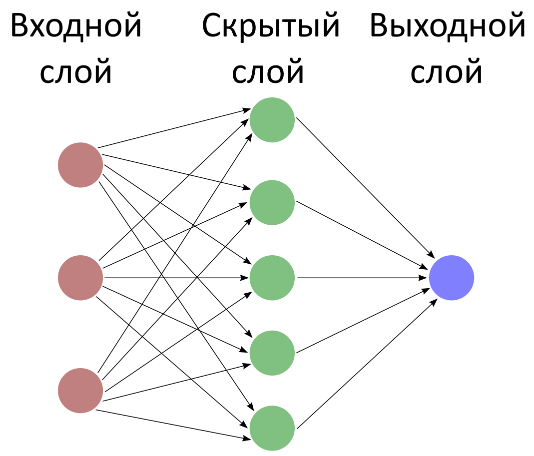


Рисунок 1.
Однослойная нейронная сеть

Решение уравнений с помощью нейронных сетей

Искусственные нейронные сети можно применить для решения дифференциальных уравнений в частных производных вида:

$$f\left(\boldsymbol{x}; u; \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_n}; \frac{\partial^2 u}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u}{\partial x_n \partial x_n}\right) = 0, \quad \boldsymbol{x} \in D, \quad D \subset \mathbb{R}^n$$

с граничными условиями:

$$b(\boldsymbol{x}, u)\big|_{\Gamma} = 0$$

В качестве меры ошибки и целевой функции можно использовать взвешенную сумму L^2 норм отклонений (невязок) по уравнению и по границе:

$$\text{loss} = \omega_f \text{loss}_f + \omega_b \text{loss}_b$$

$$\text{loss}_f = \frac{1}{|N_f|} \sum_{\boldsymbol{x} \in N_f} \left\| f\left(\boldsymbol{x}; \hat{u}; \frac{\partial \hat{u}}{\partial x_1}, \ldots, \frac{\partial \hat{u}}{\partial x_n}; \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 \hat{u}}{\partial x_n \partial x_n}\right) \right\|^2$$

$$\text{loss}_b = \frac{1}{|N_b|} \sum_{\boldsymbol{x} \in N_b} \left\| b(\boldsymbol{x}, \hat{u}) \right\|^2$$

где ω_f, ω_b — веса; $N_f = \{x_1, \ldots, x_{|N_f|}\}$, $N_b = \{x_1, \ldots, x_{|N_b|}\}$ — множества точек коллокации (тренировочная выборка) по уравнению и по границе соответственно; $|N_f|$, $|N_b|$ — количества точек; $\hat{u}$ — решение нейронной сети.

Описание задачи

Рассматриваем уравнение Гельмгольца размерности n на единичном кубе:

$$-\Delta u(\boldsymbol{x}) - u(\boldsymbol{x}) = (4\pi^2 n - 1)\cdot\prod_i^n \sin(2\pi x_i)$$

$$u|_{\Gamma} = 0$$

$$D = \{\vec{x}:\ 0 \leqslant x_i \leqslant 1,\ i = 1, \ldots, n\}$$

Правая часть уравнения выбрана таким образом, чтобы решение рассчитывалось аналитически:

$$u(\boldsymbol{x}) = \prod_i^n \sin(2\pi x_i)$$

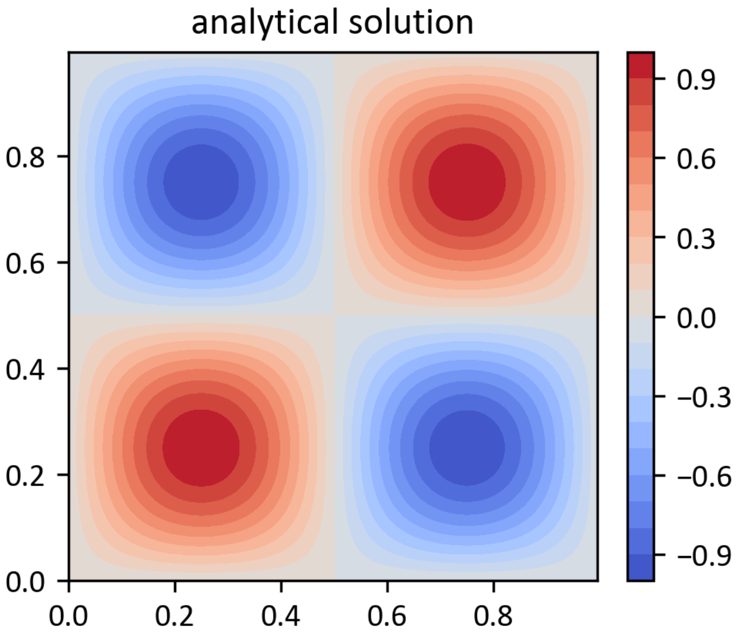


График 1. *Аналитическое решение в двумерном пространстве*

Функционал ошибки

В решаемой задаче возможно жёсткое форсирование граничного условия (см. след. слайд), поэтому функция ошибки будет состоять только из невязки по уравнению:

$$\mathrm{loss} = \mathrm{MSE}_f = \left\langle \left(\Delta\, u(\boldsymbol{x}) + u(\boldsymbol{x}) + (4\pi^2 n - 1) \cdot \prod_i^n \sin(2\pi x_i) \right)^2 \right\rangle$$

В силу того, что нейронные сети представляют собой комбинацию аналитических преобразований, для них можно рассчитать производную от выходного значения по входным параметрам. По этому принципу в данном случае производится расчёт лапласиана. PyTorch предоставляет средства для автоматического дифференцирования функций и, в частности, нейросетей.

Структура сети

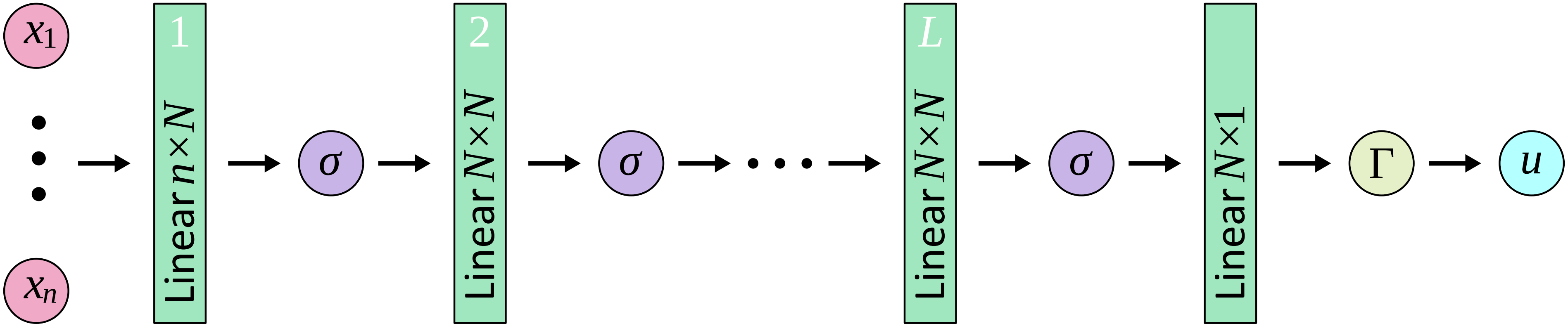


Рисунок 5. *Схема используемой нейронной сети*

Количество слоёв L, ширина слоя N, и функция активации σ подлежат подбору. На выходе сети производится форсирование граничного условия. Оно реализуется домножением выходного значения на гладкую функцию, удовлетворяющую граничным условиям:

$$\tilde{u} = u \cdot \prod_{i}^{n} (x_i - 0)(1 - x_i)$$

Генерация точек

Обучающую выборку генерируем двумя способами:

- классическим генератором равномерно распределённых случайных чисел;
- квазислучайным генератором из последовательности Соболя.

При квазислучайной генерации точек гарантируется равномерное заполнение области определения уравнения и каждого батча.

Тестовые точки будем всегда выбирать из классического равномерного распределения. Их количество всегда выбираем равным количеству точек в обучающей выборке.

Оптимизация гиперпараметров

Гиперпараметры, связанные с процессом обучения:

- $|N_f|$— количество точек в обучающей выборке;
- RNG — способ генерации точек в обучающей выборке;
- $|N_{\mathrm{batch}}|$ — размер батча;
- lr — начальная скорость обучения;
- lr scheduler — планировщик скорости обучения:
  - `None`;
  - `ExponentialLR-0.95`;
  - `ReduceLROnPlateau-0.1-10`;
  - `ReduceLROnPlateau-0.5-2`.

Гиперпараметры, связанные с архитектурой сети:

- N — ширина скрытого слоя (количество нейронов в скрытом слое);
- L — количество скрытых слоёв;
- σ — функция активации:
  - $\mathrm{ELU}(x) = \begin{cases} x, & x > 0 \\ e^x - 1, & x \le 0 \end{cases}$
  - $\mathrm{sigmoid}(x) = \dfrac{1}{1 + e^{-x}}$
  - $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$
  - $\sin(x)$
  - $\dfrac{\tan^{-1}(x)}{\pi/2}$

Вычислительные ресурсы

Обучение нейронных сетей проводилось на узлах РЦ ВЦ СПбГУ:

- ручная оптимизация — **NVIDIA Tesla P100**, 16 GB;
- автоматическая оптимизация и финальное обучение — **NVIDIA RTX A6000**, 48 GB.

RTX A6000 превосходит Tesla P100 по скорости обучения моделей примерно в 1.8 раз.

Ручная оптимизация гиперпараметров — Методика

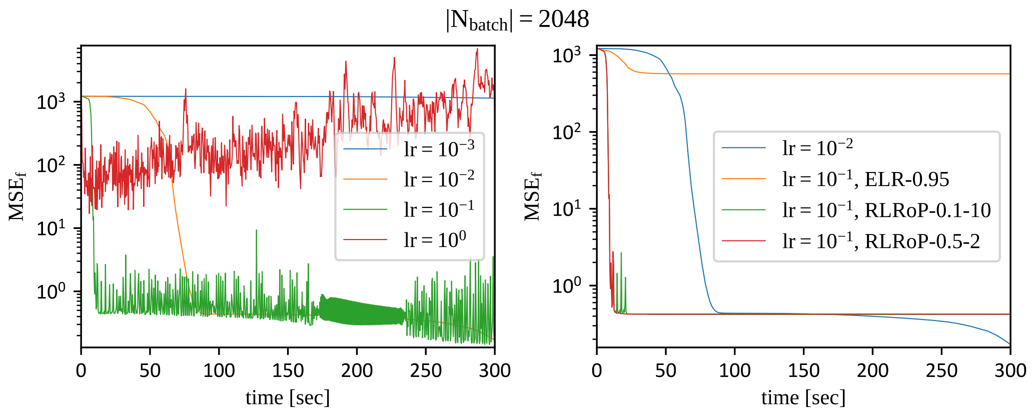
Сначала проводим исследование влияния гиперпараметров на сходимость сети в случаях 2-мерной и 5-мерной задач.

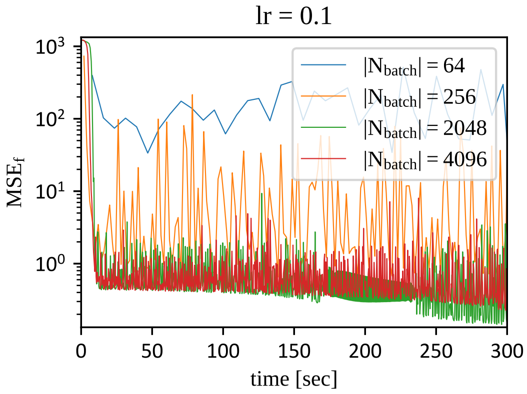
- Задаём сетку конфигураций и проводим случайный поиск по 100 точкам с одинаковым временем обучения на каждую точку.
- Стартуя с полученной оптимальной конфигурации, разбиваем гиперпараметры на группы и варьируем их по сетке. Выбираем оптимальные значения и переходим к другим гиперпараметрам. Группы выбраны следующим образом:
    - скорость обучения и размер батча;
    - количество точек и способ генерации точек;
    - ширина скрытого слоя, количество скрытых слоёв и функция активации.

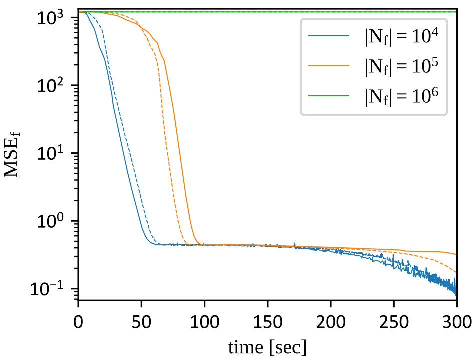
Ручная оптимизация гиперпараметров — Результаты

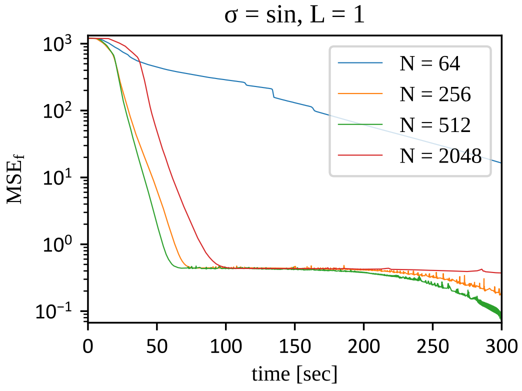
Таблица 16. *Лучшая конфигурация модели для решения 5-мерной задачи, полученная вручную*

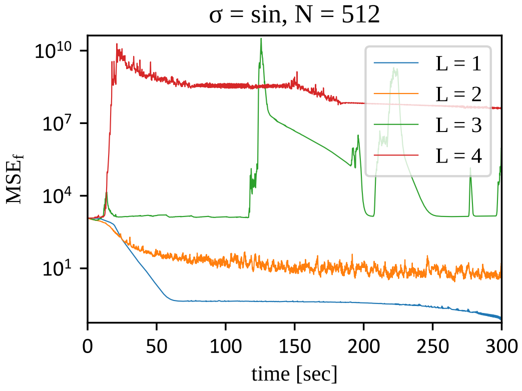
t, сек	MSE_f	$RMSE_u$	N	L	σ	$\lvert N_f \rvert$	RNG	$\lvert N_{batch} \rvert$	lr	lr scheduler
300	8.92E−02	1.52E−03	512	1	sin	10^4	квазисл.	2048	10^{-2}	None











Ручная оптимизация гиперпараметров — Выводы

- Размер батча имеет относительно низкое влияние на сходимость сети по сравнению с другими параметрами. Однако не следует выбирать размер батча слишком малым.
- При больших значениях lr целевая функция быстро понижается на первых итерациях, а затем приобретает осциллирующий вид. При малых lr сходимость медленная, но более плавная, а абсолютная точность сети, как правило, выше.
- Использование планировщика lr в большинстве случаев предпочтительнее, чем постоянное значение lr, и даёт выигрыш по скорости и абсолютной точности. Однако, неправильно подобранные параметры планировщика могут ухудшить результат.
- Увеличение числа точек коллокации не всегда улучшает скорость сходимости, а иногда даже значительно ухудшает её.
- Квазислучайный метод генерации точек превосходит случайный метод, но выигрыш, как правило, заметен при небольшом количестве точек.
- Для рассмотренной задачи лучшую аппроксимацию дают неглубокие сети с 1–2 скрытыми слоями.
- Из рассмотренных функций активации `sin` превосходит `tanh` и сильно превосходит все остальные функции.

Автоматическая оптимизация гиперпараметров

Автоматическая оптимизация гиперпараметров (**HPO**) состоит из двух основных компонентов:

- **алгоритм поиска** — выбирает точку из области гиперпараметров для рассмотрения;
- **планировщик** — заранее останавливает процесс обучения сети в случае плохой сходимости относительно предыдущих результатов.

Алгоритмы поиска

Самым базовым методом поиска является **поиск по сетке**. Такой поиск применим только для небольшого набора гиперпараметров.

Улучшение по сравнению с поиском по сетке даёт **случайный поиск**, который статистически превосходит поиск по сетке, особенно при больших размерностях области поиска.

Байесовская оптимизация — последовательный итерационный алгоритм, основанный на двух компонентах: вероятностной суррогатной модели и функции сбора. В качестве суррогатных моделей традиционно применяются гауссовы процессы и древовидный оцениватель Парзена (TPE). Методы байесовской оптимизации показывают себя лучше случайного поиска и предназначены для оптимизации достаточно ресурсоёмких функций.

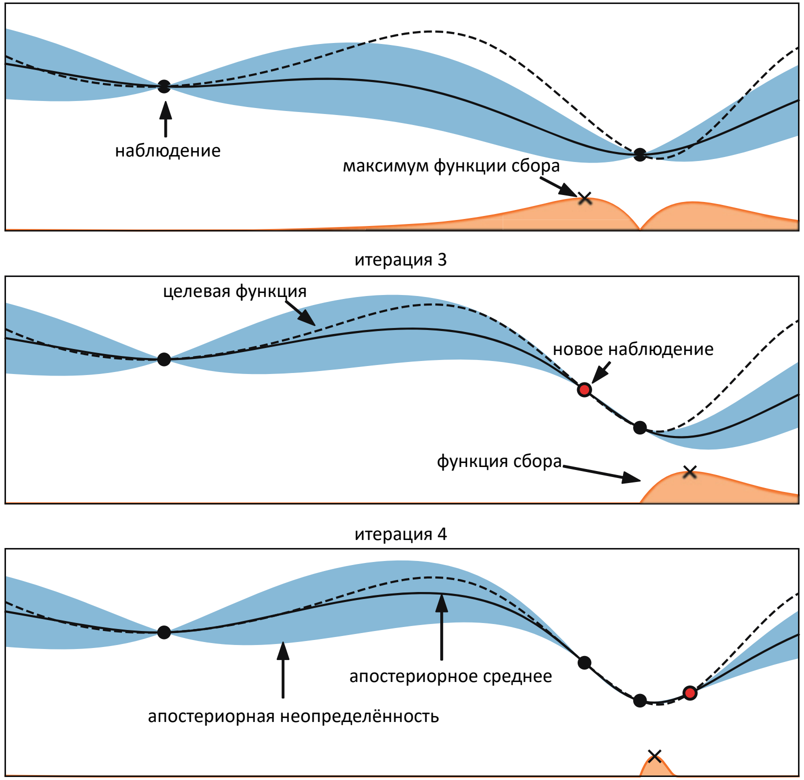


Рисунок 3. *Иллюстрация байесовской оптимизации одномерной функции*

Планировщики обучения

Медианная остановка является простейшим методом ранней остановки.

В алгоритме **последовательного деления** (**Successive Halving, SHA**) на все конфигурации выделяется доля от бюджета, по истечении которой обучение половины конфигураций останавливается. Через некоторое время прекращается обучение половины от оставшихся конфигураций. Процесс «деления» продолжается несколько раз, и в итоге только наилучшие конфигурации обучаются в течение полного времени.

HyperBand и **Asynchronous Successive Halving** (**ASHA**) являются модификациями метода SHA.

Алгоритмы поиска и планировщики обучения могут работать в комбинации независимо друг от друга. **BOHB** был разработан как попытка объединить байесовский поиск и HyperBand в единый эффективный алгоритм.

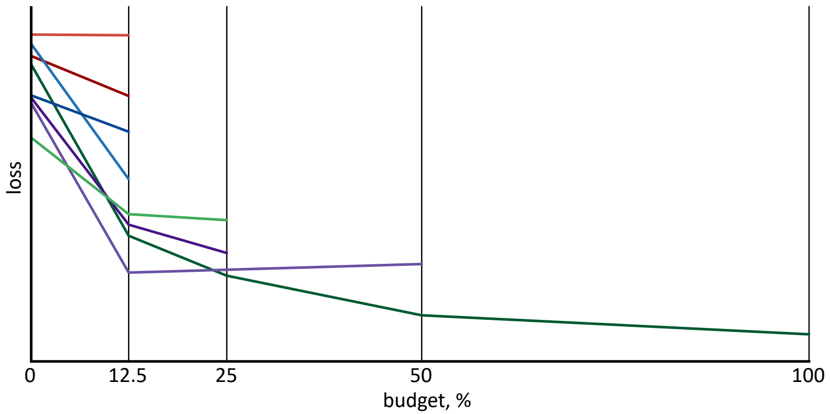


Рисунок 4. *Иллюстрация метода последовательного деления*

Программные пакеты

Для автоматической оптимизации гиперпараметров используем фреймворк **Ray Tune**. Ray Tune предоставляет единый интерфейс для работы с множеством популярных HPO-пакетов:

- **алгоритмы поиска**: поиск по сетке, случайный поиск, Ax, BayesOptSearch, BOHB, BlendSearch, CFO, Dragonfly, HEBO, HyperOpt, Optuna, SigOpt, Scikit-Optimize, ZOOpt;
- **планировщики**: ASHA, HyperBand, Median Stopping Rule, Population Based Training (PBT), Population Based Bandits (PB2), BOHB.

Из алгоритмов поиска рассматриваются **случайный поиск** и **HyperOpt**; из планировщиков рассматривается **ASHA**. Также рассматривается комбинированный алгоритм **BOHB**.

Сравнение алгоритмов оптимизации гиперпараметров

Таблица 17. *Область поиска и ограничение по времени*

N	`qlograndint(64, 2048, 32)`
L	`randint(1, 5)`
σ	`ELU, sigmoid, tanh, sin, atan`
$\lvert N_f \rvert$	$10^4, 10^5, 10^6$
$\lvert N_{\text{batch}} \rvert$	`qlograndint(64, 32768, 32)`
lr	`qloguniform(1e-4, 1, 1e-4)`
lr scheduler	`None, ExponentialLR-0.95, ReduceLROnPlateau-0.1-10, ReduceLROnPlateau-0.5-2`
t	300 сек
T	8 часов

Сравнение алгоритмов оптимизации гиперпараметров

Таблица 18. *Лучшие конфигурации модели, полученные алгоритмами HPO*

$\lvert N_{cfg}\rvert$	№	i	t, сек	MSE_f	$RMSE_u$	N	L	σ	$\lvert N_f\rvert$	$\lvert N_{batch}\rvert$	lr	lr scheduler
Случайный поиск												
96	5	107	300.1	5.0E−02	6.3E−04	1728	1	sin	10^5	160	0.355	ReduceLROnPlateau-0.5-2
	10	697	300.0	4.3E−01	1.3E−03	96	1	sin	10^5	1056	0.217	ReduceLROnPlateau-0.1-10
	21	6	304.9	3.2E+00	1.3E−02	1088	3	sin	10^6	128	0.001	None
Случайный поиск + ASHAScheduler												
1593	364	550	300.1	1.2E−02	2.6E−04	1216	1	sin	10^5	1760	0.794	ReduceLROnPlateau-0.5-2
	581	1039	300.1	2.8E−02	2.7E−04	704	1	sin	10^5	4608	0.955	ReduceLROnPlateau-0.1-10
	5	106	300.1	4.7E−02	4.4E−04	1728	1	sin	10^5	160	0.355	ReduceLROnPlateau-0.5-2
HyperOptSearch												
96	90	277	300.2	3.3E−01	1.7E−03	768	2	sin	10^5	928	0.008	ReduceLROnPlateau-0.5-2
	76	1130	300.1	4.3E−01	1.3E−03	416	1	sin	10^5	2816	0.215	ReduceLROnPlateau-0.5-2
	81	250	300.2	5.3E−01	2.1E−03	928	2	sin	10^5	1408	0.002	ReduceLROnPlateau-0.5-2
HyperOptSearch + ASHAScheduler												
1053	817	624	300.1	**1.1E−02**	**1.7E−04**	928	1	sin	10^5	2016	0.996	ReduceLROnPlateau-0.5-2
	548	696	300.1	1.1E−02	2.3E−04	736	1	sin	10^5	1824	0.857	ReduceLROnPlateau-0.5-2
	607	690	300.1	1.1E−02	1.7E−04	704	1	sin	10^5	1728	0.992	ReduceLROnPlateau-0.5-2
BOHB												
1108	820	1695	200.0	4.1E−01	1.4E−03	544	2	sin	10^4	736	0.014	ReduceLROnPlateau-0.1-10
	223	326	219.5	4.2E−01	1.2E−03	128	1	sin	10^5	736	0.434	ReduceLROnPlateau-0.1-10
	1059	2561	219.0	4.3E−01	1.2E−03	160	1	sin	10^4	736	0.794	ReduceLROnPlateau-0.1-10

Автоматическая оптимизация гиперпараметров — 2-мерная задача

Таблица 19. *Область поиска и ограничение по времени для 2-мерной задачи*

N	`qlograndint(64, 2048, 32)`
L	`randint(1, 5)`
σ	`ELU, sigmoid, tanh, sin, atan`
$\|N_f\|$	10^2, 10^3, 10^4
$\|N_{\text{batch}}\|$	`qlograndint(64, 32768, 32)`
lr	`qloguniform(1e-4, 1, 1e-4)`
lr scheduler	`None, ExponentialLR-0.95, ReduceLROnPlateau-0.1-10, ReduceLROnPlateau-0.5-2`
t	30 сек
T	1 час

Таблица 20. *Лучшие конфигурации модели, полученные для 2-мерной задачи*

$\|N_{\text{cfg}}\|$	№	i	t, сек	MSE_f	RMSE_u	N	L	σ	$\|N_f\|$	$\|N_{\text{batch}}\|$	lr	lr scheduler
665	457	1306	30.0	5.6E−05	1.4E−05	416	2	sin	1000	640	0.028	ReduceLROnPlateau-0.5-2
	594	1286	30.0	7.8E−05	1.1E−05	672	2	sin	1000	384	0.015	ReduceLROnPlateau-0.5-2

Автоматическая оптимизация гиперпараметров — 2-мерная задача

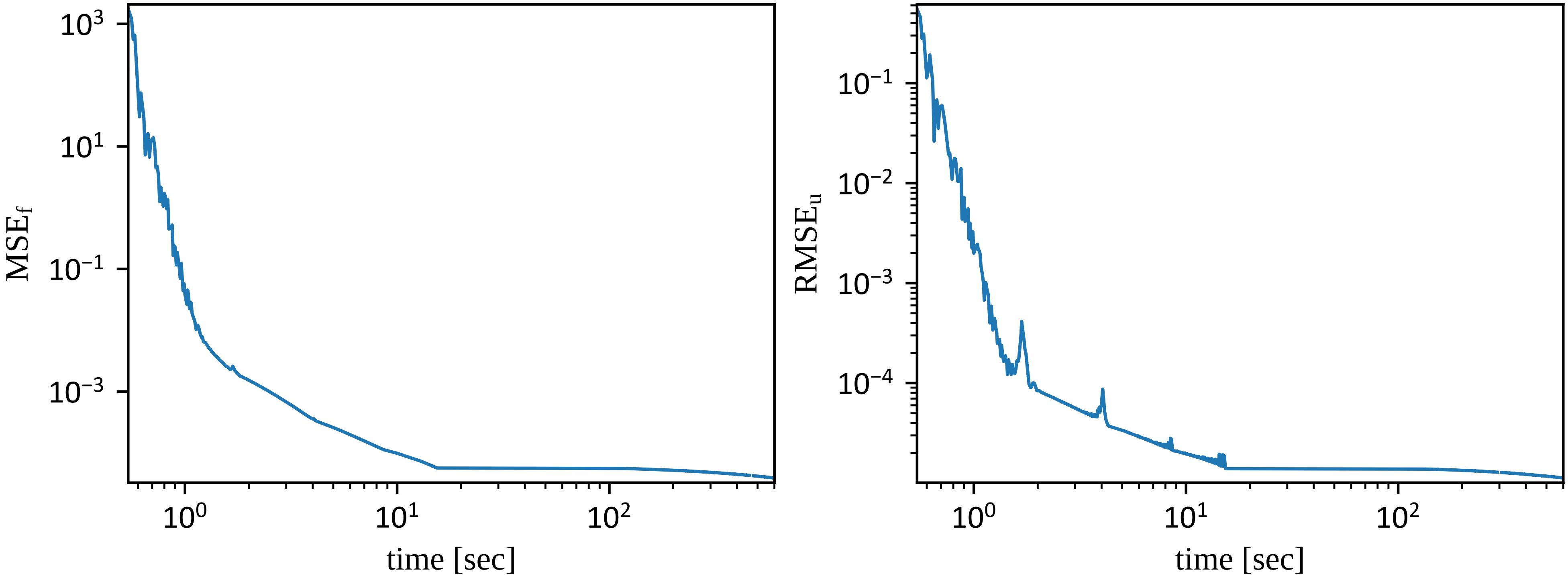


График 12.

Зависимость невязки по уравнению и невязки по решению от времени лучшей конфигурации для 2-мерной задачи

Автоматическая оптимизация гиперпараметров — 2-мерная задача

Конечные значения невязок:

$MSE_f = 3.885E{-}05$

$RMSE_u = 1.125E{-}05$

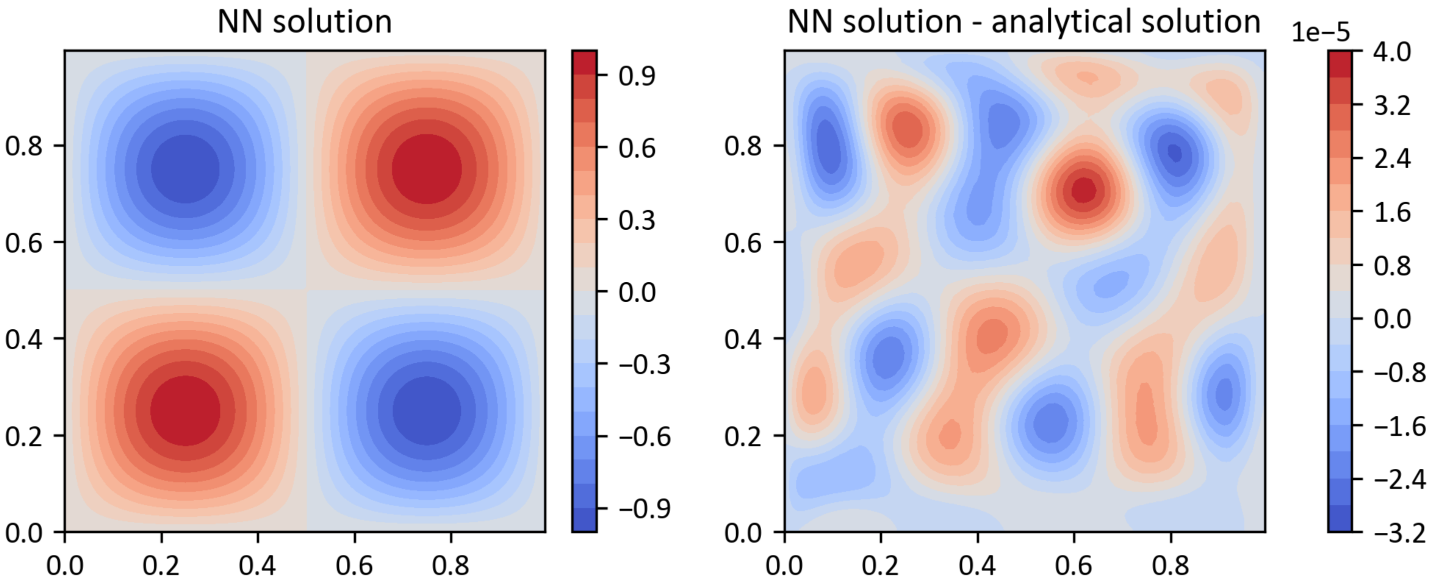


График 13.
Лучшее решение 2-мерной задачи и его отклонение от аналитического решения

Автоматическая оптимизация гиперпараметров — 5-мерная задача

Таблица 21. *Область поиска и ограничение по времени для 5-мерной задачи*

N	`qlograndint(64, 2048, 32)`
L	`randint(1, 5)`
σ	`ELU, sigmoid, tanh, sin, atan`
$\lvert N_f \rvert$	10^4, 10^5, 10^6
$\lvert N_{\text{batch}} \rvert$	`qlograndint(64, 32768, 32)`
lr	`qloguniform(1e-4, 1, 1e-4)`
lr scheduler	`None, ExponentialLR-0.95, ReduceLROnPlateau-0.1-10, ReduceLROnPlateau-0.5-2`
t	300 сек
T	8 часов

Таблица 22. *Лучшие конфигурации модели, полученные для 5-мерной задачи*

$\lvert N_{\text{cfg}} \rvert$	№	i	t, сек	MSE_f	RMSE_u	N	L	σ	$\lvert N_f \rvert$	$\lvert N_{\text{batch}} \rvert$	lr	lr scheduler
1053	817	624	300.1	1.1E−02	1.7E−04	928	1	sin	105	2016	0.996	ReduceLROnPlateau-0.5-2
	548	696	300.1	1.1E−02	2.3E−04	736	1	sin	105	1824	0.857	ReduceLROnPlateau-0.5-2

Автоматическая оптимизация гиперпараметров — 5-мерная задача

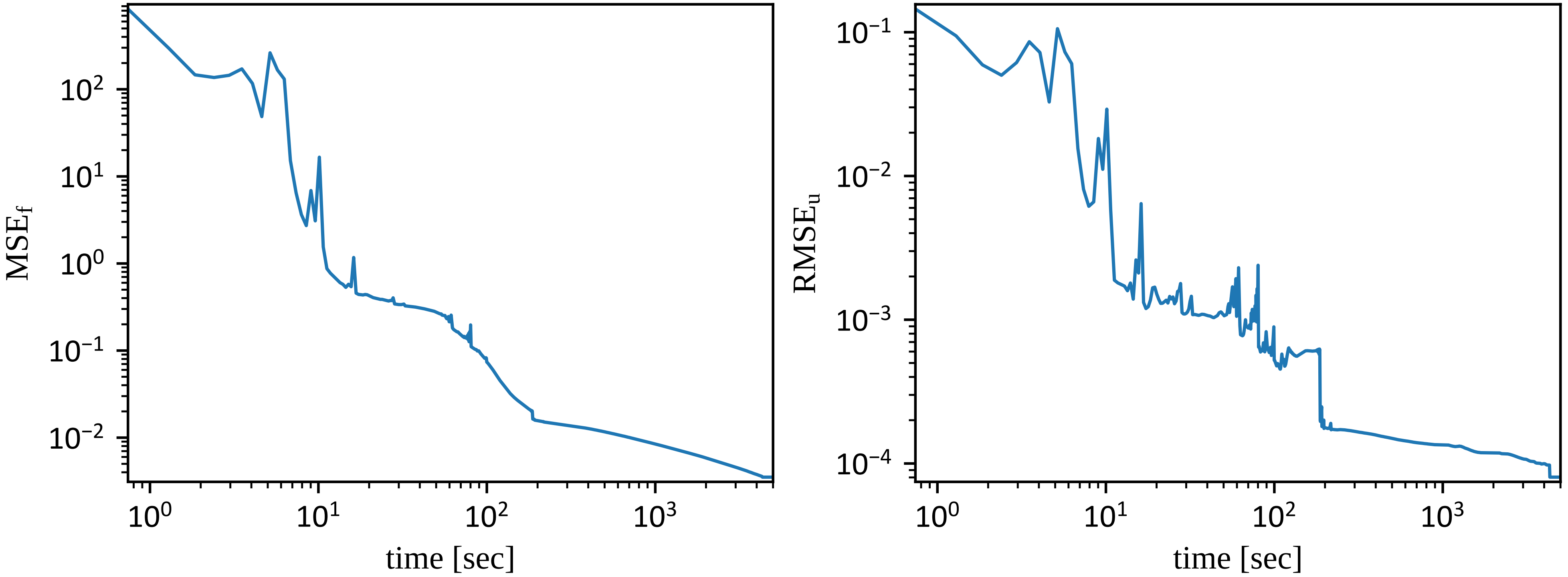


График 14.

Зависимость невязки по уравнению и невязки по решению от времени лучшей конфигурации для 5-мерной задачи

Автоматическая оптимизация гиперпараметров — 5-мерная задача

Конечные значения невязок:

$MSE_f = 3.518E{-}03$

$RMSE_u = 8.036E{-}05$

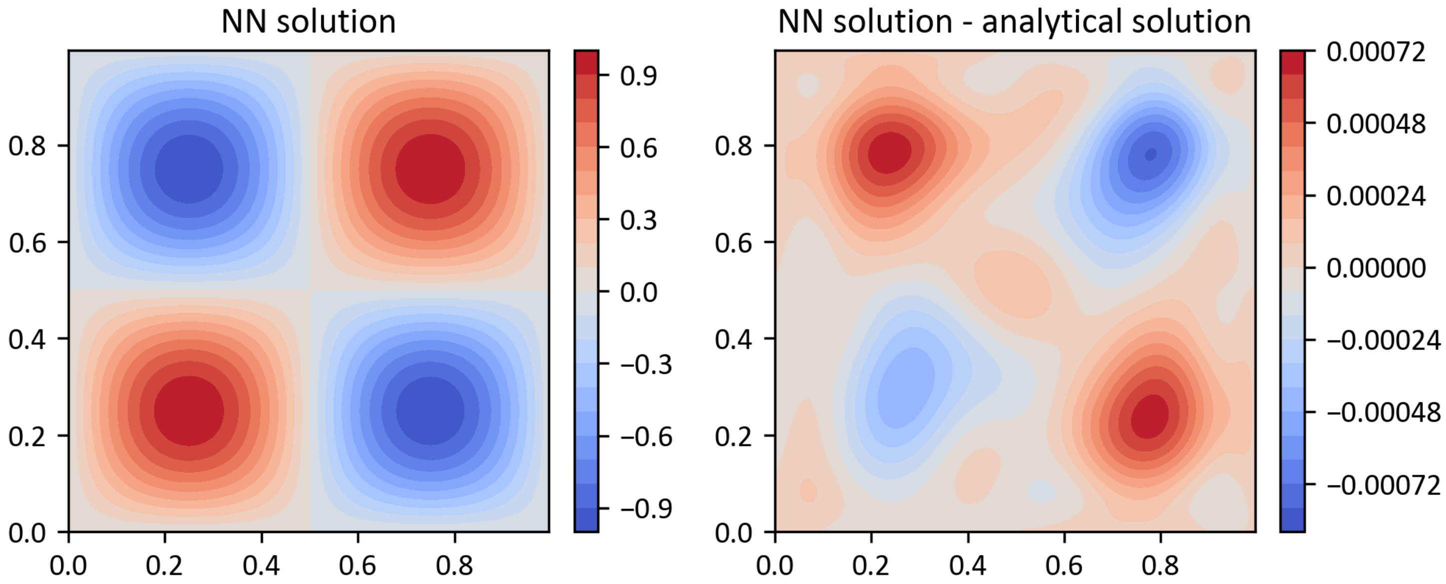


График 15.
Лучшее решение 5-мерной задачи и его отклонение от аналитического решения

Автоматическая оптимизация гиперпараметров — 8-мерная задача

Таблица 23. *Область поиска и ограничение по времени для 8-мерной задачи*

N	`qlograndint(64, 2048, 32)`
L	`randint(1, 5)`
σ	`ELU, sigmoid, tanh, sin, atan`
$\lvert N_f \rvert$	10^6, 10^7, 10^8
$\lvert N_{batch} \rvert$	`qlograndint(64, 32768, 32)`
lr	`qloguniform(1e-4, 1, 1e-4)`
lr scheduler	`None, ExponentialLR-0.95, ReduceLROnPlateau-0.1-10, ReduceLROnPlateau-0.5-2`
t	1800 сек
T	48 часов

Таблица 24. *Лучшие конфигурации модели, полученные для 8-мерной задачи*

$\lvert N_{cfg} \rvert$	№	i	t, сек	MSE_f	$RMSE_u$	N	L	σ	$\lvert N_f \rvert$	$\lvert N_{batch} \rvert$	lr	lr scheduler
313	260	60.9	1802.1	3.3E+01	1.5E−02	2016	1	sin	10^6	160	0.068	ExponentialLR-0.95
	262	60.2	1802.2	3.3E+01	1.5E−02	2016	1	sin	10^6	160	0.071	ExponentialLR-0.95

Автоматическая оптимизация гиперпараметров — 8-мерная задача

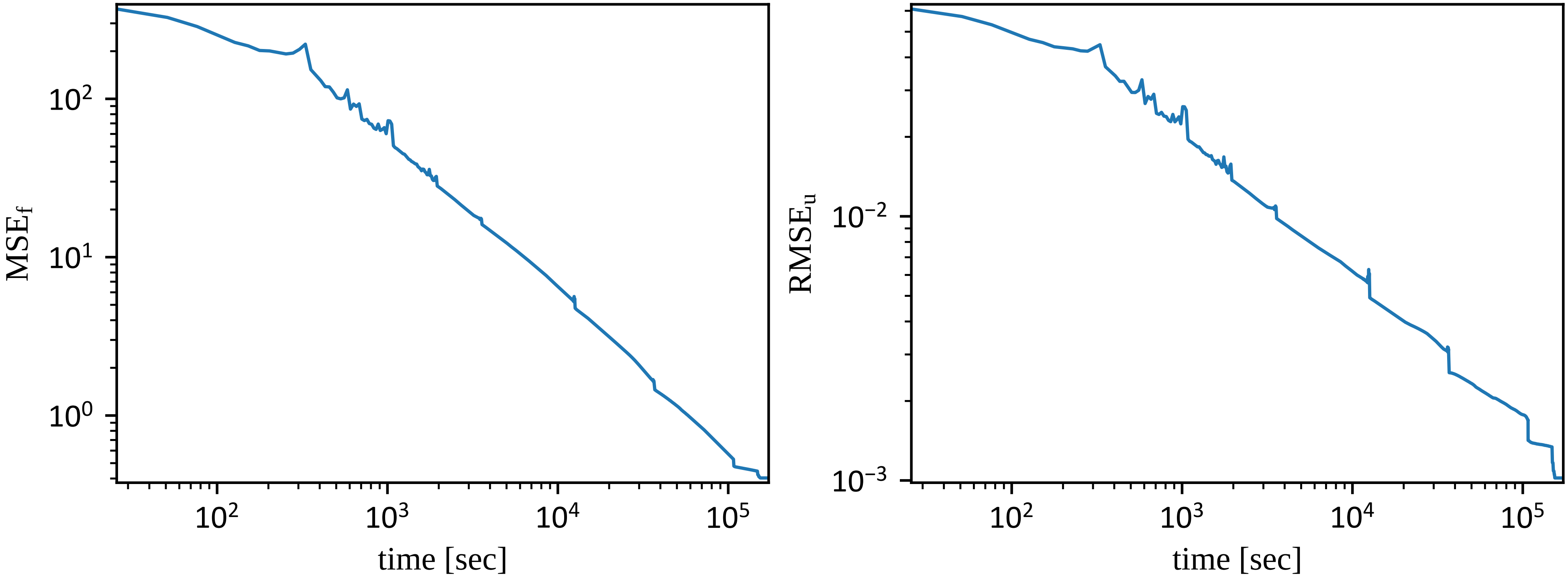


График 16.

Зависимость невязки по уравнению и невязки по решению от времени лучшей конфигурации для 8-мерной задачи

Автоматическая оптимизация гиперпараметров — 8-мерная задача

Конечные значения невязок:

$\mathrm{MSE}_f = 4.030\mathrm{E}{-01}$

$\mathrm{RMSE}_u = 1.022\mathrm{E}{-03}$

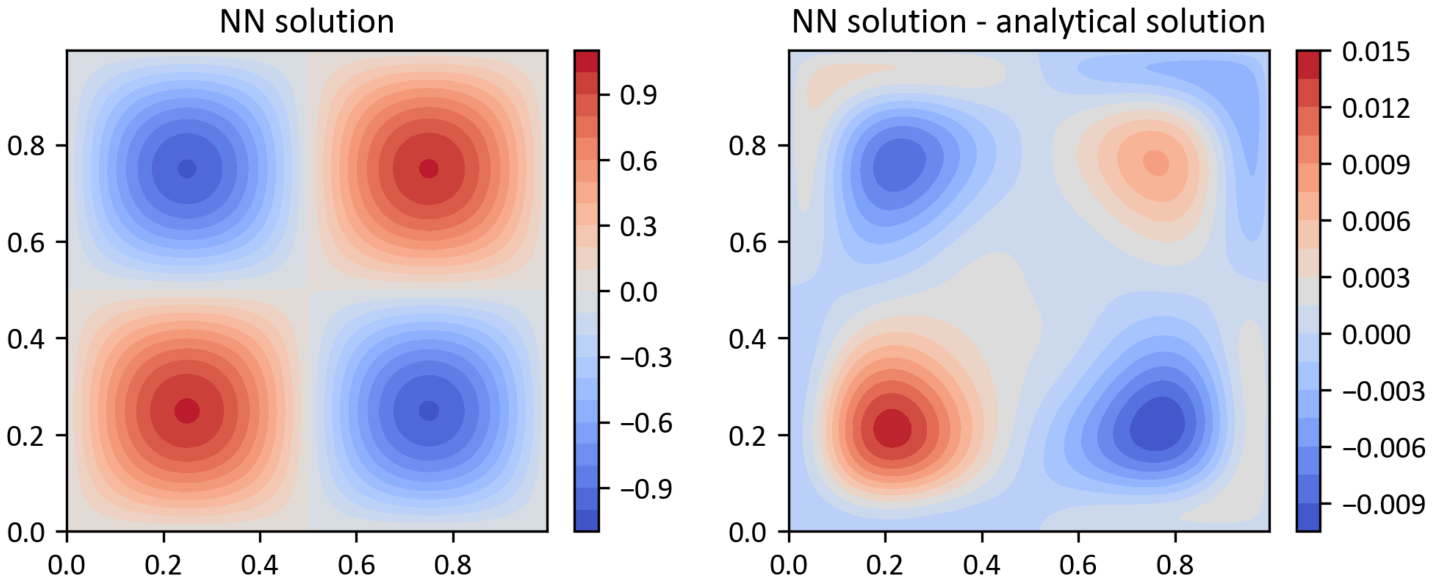


График 17.
Лучшее решение 8-мерной задачи и его отклонение от аналитического решения

Выводы

- В ходе работы удалось эффективно использовать средства автоматической оптимизации гиперпараметров (HPO) для подбора конфигурации модели, аппроксимирующей уравнение Гельмгольца. Использование алгоритма ранней остановки позволяет рассмотреть гораздо больше конфигураций, и целесообразно в первую очередь применять его, а для большей точности можно подключить и байесовский алгоритм поиска. Из рассмотренных алгоритмов HPO наилучший результат был получен при комбинации HyperOptSearch + ASHAScheduler.
- Посредством автоматической оптимизации гиперпараметров было получено более точное решение, чем при ручной оптимизации.
- В работе продемонстрирована возможность решения уравнения Гельмгольца посредством нейронных сетей в пространствах больших размерностей (до 8). Однако, при увеличении размерности задачи быстро растут потребности в вычислительных ресурсах, что смягчается возможностью автоматизации процесса.