Skolkovo Institute of Science and Technology

MASTER'S THESIS

# Fantastic grants and where to find them

Master's Educational Program: Startups, memes and bullshitting


Student________________________________

Josef Svejk

Startups, memes and bullshitting

June 18, 2019

Research Advisor:__________________________

Dmitriy L. Kishmish

Associate Professor

Co-Advisor:____________________________

Kozma P. Prutkov

Associate Professor


Moscow 2019

Skolkovo Institute of Science and Technology

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

# Фантастические гранты и их места обитания

Магистерская образовательная программа: Стартапов, мемов и макарон

Студент____________________

Иозеф Швейк
Стартапов, мемов и макарон
Июнь 18, 2019

Научный руководитель:____________________

Дмитрий Л. Кишмиш
Профессор

Со-руководитель:____________________

Козьма П. Прутков
Профессор

Москва 2019

# Fantastic grants and where to find them

Josef Svejk

Submitted to the Skolkovo Institute of Science and Technology
on June 18, 2019

## Abstract

As any dedicated reader can clearly see, the Ideal of practical reason is a representation of, as far as I know, the things in themselves; as I have shown elsewhere, the phenomena should only be used as a canon for our understanding. The paralogisms of practical reason are what first give rise to the architectonic of practical reason. As will easily be shown in the next section, reason would thereby be made to contradict, in view of these considerations, the Ideal of practical reason, yet the manifold depends on the phenomena. Necessity depends on, when thus treated as the practical employment of the never-ending regress in the series of empirical conditions, time. Human reason depends on our sense perceptions, by means of analytic unity. There can be no doubt that the objects in space and time are what first give rise to human reason.

Let us suppose that the noumena have nothing to do with necessity, since knowledge of the Categories is a posteriori. Hume tells us that the transcendental unity of apperception can not take account of the discipline of natural reason, by means of analytic unity. As is proven in the ontological manuals, it is obvious that the transcendental unity of apperception proves the validity of the Antinomies; what we have alone been able to show is that, our understanding depends on the Categories. It remains a mystery why the Ideal stands in need of reason. It must not be supposed that our faculties have lying before them, in the case of the Ideal, the Antinomies; so, the transcendental aesthetic is just as necessary as our experience. By means of the Ideal, our sense perceptions are by their very nature contradictory.

Research Advisor:
Name: Dmitriy L. Kishmish
Degree: Professor of sour soup
Title: Associate Professor

Co-Advisor:
Name: Kozma P. Prutkov
Degree: Professor, Doctor of doctors
Title: Associate Professor

## Фантастические гранты и их места обитания

Иозеф Швейк

Представлено в Сколковский институт науки и технологий
Июнь 18, 2019

## Реферат

Не без некоторого колебания решился я избрать предметом настоящей лекции философию и идеал анархизма. Многие до сих пор еще думают, что анархизм есть не что иное, как ряд мечтаний о будущем или бессознательное стремление к разрушению всей существующей цивилизации. Этот предрассудок привит нам нашим воспитанием, и для его устранения необходимо более подробное обсуждение вопроса, чем то, которое возможно в одной лекции. В самом деле, давно ли — всего несколько лет тому назад — в парижских газетах пресерьезно утверждалось, что единственная философия анархизма — разрушение, а единственный его аргумент — насилие.

Тем не менее об анархистах так много говорилось за последнее время, что некоторая часть публики стала наконец знакомиться с нашими теориями и обсуждать их, иногда даже давая себе труд подумать над ними; и в настоящую минуту мы можем считать, что одержали победу по крайней мере в одном пункте: теперь уже часто признают, что у анархиста есть некоторый идеал — идеал, который даже находят слишком высоким и прекрасным для общества, не состоящего из одних избранных.

Но не будет ли, с моей стороны, слишком смелым говорить о философии в той области, где, по мнению наших критиков, нет ничего, кроме туманных видений отдаленного будущего? Может ли анархизм претендовать на философию, когда ее не признают за социализм вообще?

Научный руководитель:
Имя: Дмитрий Л. Кишмиш
Ученое звание, степень: Профессор кислых щей
Должность: Профессор

Со-руководитель:
Имя: Козьма П. Прутков
Ученое звание, степень: Профессор, Доктор докторов
Должность: Профессор

# Acknowledgments

This is the acknowledgements section. You should replace this with your own acknowledgements.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Solving differential equations

Before starting talking about solving a single partial differential equation (PDE) or system of PDEs and about neural networks need to clearly understand what are the existing methods for this problem, how they work, what the strong and weak sides, which facts influence the quality of the solution.

## 1.1 Function approximation

### 1.1.1 Что-то про аппроксимацию и регуляризации в кратце

Let's start from supervised learning and suppose the set of pairs is given:

$$D = \{x^i, y^i\}_{i=1}^N$$

where

$$y^i = f(x^i) + \epsilon$$

In other words, here is presented the dataset of function values in some nodes. In general case not important the dimension of $x$ and $y$, that is why the previous relation for $y$ can be rewritten as:

$$f : A \to B, \quad A \in R^n, B \in R^m$$

For simplicity, let $n = 1$, $m = 1$, for the other cases the same way.

There are a lot of ways to build the approximation, for example using linear model, Linear regression, or using more advanced techniques, Ridge regression or Neural Networks (NN). For example, Linear regression:

$$\hat{y}^j = \beta_0 + \sum_{i=1}^{n} \beta_i x_i^j = \beta_0 + \beta_1 x^j \tag{1.1}$$

and the main goal is to estimate the coefficients $a_0$ and $a_1$. Here $n$ is the dimension of the

$A$ space. If the dimension of $A$ is more than 1, the matrix form is more suitable for (1.1):

$$\hat{y}^j = \beta_0 + \sum_{i=1}^{n} \beta_i x_i^j = x^T \beta \implies Y = X\beta \tag{1.2}$$

In the general case, (1.2) can be rewritten as:

$$\hat{y}^j = \beta_0 + \sum_{i=1}^{K} \beta_i \phi_i(x_i^j) \quad \text{or} \quad Y = Z\beta, \text{where } Z_i^j = \phi_i(x_i^j) \tag{1.3}$$

where the functions $\phi_j$ are predefined earlier depends on the specificity of the problem. $K$ is the count of the predefined functions.

For the regression problem and for coefficients estimation the quality function or loss function is needed to be defined.

$$R(x) = \hat{y} - y, \quad R(x^i) = R^i = \hat{y}^i - y^i \tag{1.4}$$

residual at $x^i$. The main goal is to minimize the sum of residuals:

$$\sum_{x^i \in X} R(x) \to min, \quad \text{or} \sum_{x^i \in X} g(R(x)) \to min$$

, $g$ is monotonic function - loss function. The most widely used loss function for this type of problem is the mean squared error or $R^2$ score. In this work, the mean squared error will be used:

$$\mathcal{L} = \frac{1}{N}\sqrt{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2} = \frac{1}{N}\sqrt{\sum_{i=1}^{N} (R^i)^2} \tag{1.5}$$

where $y^i$ is the function value at $x^i$ from the dataset and $\hat{y^i}$ predicted from the model.

For the estimate, the coefficients use the least-squares method for (1.2):

$$\frac{\partial \mathcal{L}}{\partial a_i} = \frac{\partial}{\partial a_i} \frac{1}{N}\sqrt{\sum_{i=1}^{N} (R(x^i))^2}$$

The considered way very powerful for estimation coefficients, for analysis and can be effectively solved via linear algebra instruments. Using statistical methods the number of needed functions and their values estimates with their confidence intervals. The problem can arise, when the possibility to calculate the derivatives is absent or the extremum of the loss function is not unique.

### 1.1.2 Что то про линейную регрессию

From (1.2) linear model is:

$$Y = X\beta$$

$\beta$ - unknown parameters. The residual for this model is: Now, just substitute the residual to loss function (1.5):

$$\mathcal{L} = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(R^i)^2} \Leftrightarrow \|Y - X\beta\|^2 \rightarrow min$$

$$\|Y - X\beta\|^2 = (Y - X\beta)^T (Y - X\beta) = Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T X\beta \tag{1.6}$$

And compute $\dfrac{\partial \mathcal{L}}{\partial \beta}$:

$$\frac{\partial}{\partial \beta}\left[Y^T Y - Y^T X\beta - \beta^T X^T Y + \beta^T X^T X\beta\right] \implies X^T Y = X^T X\beta\beta = \left(X^T X\right)^{-1} X^T Y$$

The last operation was very dangerous in sense when the inverse matrix doesn't exist or ill-conditioned. For example, if the determinant of matrix $X^T X$ doesn't exist what should do? Or $X^T X$ is ill-conditioned? The answer is to apply the special techniques to avoid it - regularization. Look at the (1.6) and add the additional term [15]:

$$\|Y - X\beta\|^2 + \lambda\|\beta\|^2 \implies \beta = \left(X^T X + \lambda I\right)^{-1} X^T Y \tag{1.7}$$

From [15] implies fact, that $X^T X + \lambda I$ is not singular matrix and in fact has better condition number than $X^T X$. Moreover, there are a lot of regularization techniques [13], [23], [3], [17], [20]. It is only one simple example of problems, arises during the approximation process. By the way, more powerful methods exist, avoiding some problems: Random Forest, Gradient Boosting citebishop, Neural Networks [12].

**Impact of the regularization** Let $y = f(x) = kx + \epsilon, k = 10$. At the fig. 1.1 seen, that regularization impact is big. After applying linear models ([13], [23], [17]) for this problem with different regularization methods was got a different results. The RLAD method estimate the $\hat{k} = 10.498$, Ridge method $\hat{k} = 9.109$ and Lasso - $\hat{k} = 9.604$. Results slightly different because the key difference is using different norms for regularization. It is an important fact for the next work, where more complex regression models will be used.

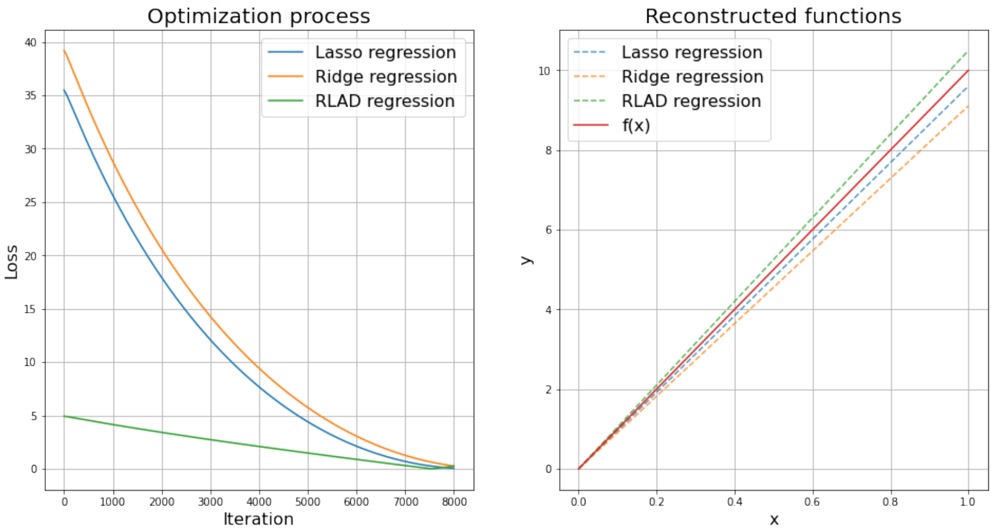


Figure 1.1: Comparison of different regularizations

## 1.2 Expansion the functions into the functional series

Linear regression looks like the function expansion, in some sense, into the series of predefined functions (1.3). If let $\phi_i = cos$ then the linear regression transforms to cosine expansion, but without orthogonality criterion. Instead of cosines can be used any functions, Chebyshev polynomials or Legendre polynomials. Let's talk about each function separately.

### 1.2.1 Fourier series

Instead of arbitrary $\phi_i$ substitute $e^{i\pi k}$ to (1.3):

$$S_K(x) = \beta_0 + \sum_{i=1}^{K} \beta_i e^{i\pi kx} \implies S(x) = a_0 + \sum_{i=1}^{K} (a_i cos(\pi kx) + b_i sin(\pi kx))$$

This set of functions is orthogonal on the considered interval:

$$\int_{-\pi}^{\pi} e^{i\pi kx} e^{i\pi lx} = 2\pi\delta_k^l$$

The estimation of $a_n$ and $b_n$ for some function $f$ is a straightforward process, first of all, define the residual for this expansion - $R(x) = f(x) - S_K(x)$, after that, integrate the squared residual over the domain and use the least-squares method:

$$
\begin{aligned}
\mathcal{L} = \int_{\Omega} R(x)^2 d\Omega = \int_{\Omega} \left(f(x) - S_K(x)\right)^2 d\Omega = \\
= \int_{\Omega} f(x)^2 d\Omega - 2\int_{\Omega} S_K(x) f(x) d\Omega + \int_{\Omega} S_K(x)^2 d\Omega = \\
= \int_{\Omega} f(x)^2 d\Omega - 2\int_{\Omega} \left[a_0 + \sum_{i=1}^{K} \left(a_i cos(\pi kx) + b_i sin(\pi kx)\right)\right] f(x) d\Omega + \\
+ \int_{\Omega} \left[a_0 + \sum_{i=1}^{K} \left(a_i cos(\pi kx) + b_i sin(\pi kx)\right)\right]^2 d\Omega
\end{aligned}
$$

And the derivates of the integrated residual:

$$
\begin{cases}
\dfrac{\partial}{\partial a_n}\mathcal{L} = \dfrac{\partial \mathcal{L}}{\partial S_K(x)} \dfrac{\partial S_K(x)}{\partial a_n} = -2\displaystyle\int_{\Omega} f(x) cos(\pi kx) d\Omega + -2\int_{\Omega} S_K(x) cos(\pi kx) d\Omega \\
\dfrac{\partial}{\partial b_n}\mathcal{L} = \dfrac{\partial \mathcal{L}}{\partial S_K(x)} \dfrac{\partial S_K(x)}{\partial b_n} = -2\displaystyle\int_{\Omega} f(x) sin(\pi kx) d\Omega + -2\int_{\Omega} S_K(x) sin(\pi kx) d\Omega \\
\dfrac{\partial}{\partial a_0}\mathcal{L} = \dfrac{\partial \mathcal{L}}{\partial S_K(x)} \dfrac{\partial S_K(x)}{\partial a_0} = -2\displaystyle\int_{\Omega} f(x) d\Omega + 2|\Omega| a_0
\end{cases}
\tag{1.8}
$$

The main part of the calculations is absent and provided here [2]. In addition, there is a convergence analysis of the coefficients, in sense of pointwise, and $L_2$ norm. The final result for the coefficients is:

$$
\begin{cases}
a_0 = \dfrac{\displaystyle\int_{\Omega} f(x) d\Omega}{2|\Omega|} \\
a_n = \dfrac{\displaystyle\int_{\Omega} f(x) cos(\pi kx) d\Omega}{|\Omega|} \\
b_n = \dfrac{\displaystyle\int_{\Omega} f(x) sin(\pi kx) d\Omega}{|\Omega|}
\end{cases}
\tag{1.9}
$$

**Example of function expansion into the Fourier series** Consider the function $y = sin(x) + x$ and at the fig. 1.2 the results. It can be seen that with a relatively small amount of the terms the approximation good. With 3 terms the approximation error[1] is 0.565, 5 terms - 0.005 and 10 terms

[1]Here, the approximation error is the loss function and concrete - mean squared error between the known values and Fourier expansion. There is a pointwise loss value, where the error calculation includes the finite number of nodes and integral loss value, where the residual integrates over the all domain.

is 0.002.

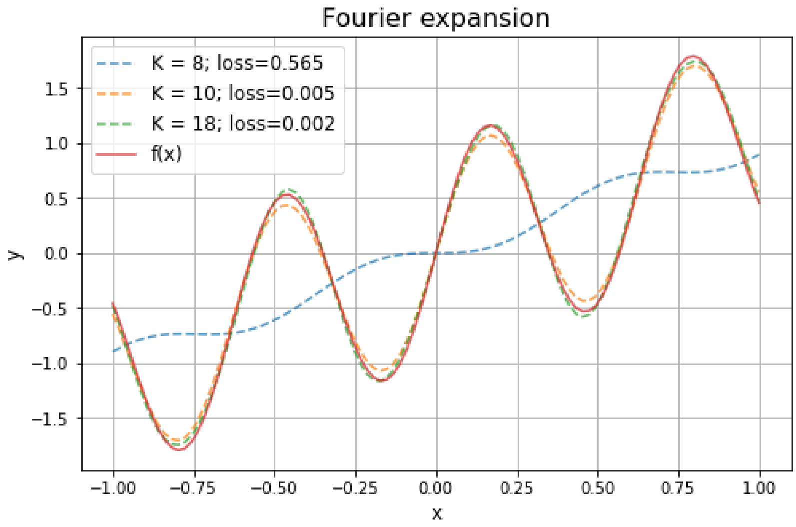


Figure 1.2: Example of function expansion into the Fourier series with 3, 10, 18 terms

**The strong sides of Fourier expansion**

There are two important theorems helps to use Fourier expansion for construction of the future approximation:

**Theorem 1.1** *If $f$ belongs to $L^2([-\pi, \pi])$ then $S_k$ converges to $f$ in $L^2([-\pi, \pi])$, that is, $\|S_K - f\|_2$ converges to 0 as $N \to \infty$.*

**Theorem 1.2** *If $f$ belongs to $C^1([-\pi, \pi])$ then $S_k$ converges to $f$ uniformly (and hence also point-wise).*

The proofs of theorems well provided here [2]. And an additional fact, Fourier coefficients of any integrable function tend to zero.

### 1.2.2 Chebyshev polynomials and series

Again, instead of using trigonometric polynomials, try to apply another system of orthogonal polynomials - Chebyshev polynomials and expansion into the Chebyshev series. The Chebyshev polynomials can be defined as the recurrent sequence:

$$T_0(x) = 1, T_1(x) = x, \ldots, T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \tag{1.10}$$

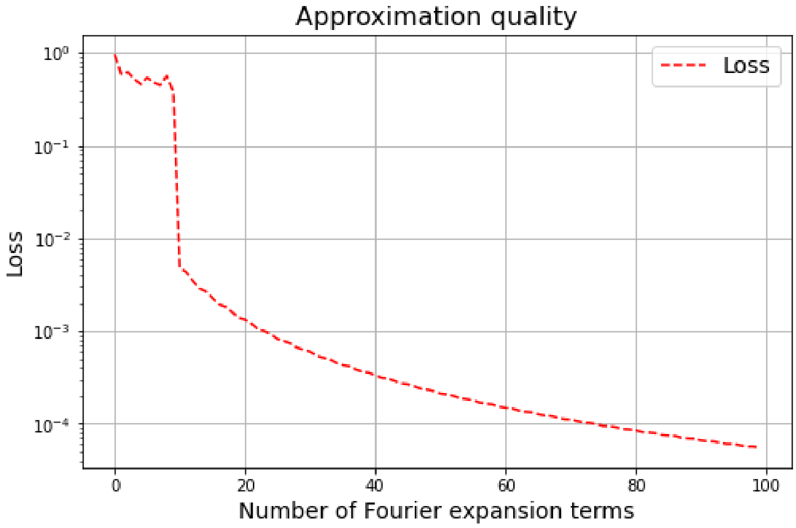


Figure 1.3: Illustration of the theorems 1.1, 1.2 for the function from the previous example

This polynomials are orthogonal with weight $w(x) = \dfrac{1}{\sqrt{1-x^2}}$[19]:

$$\int_{-1}^{1} T_k(x)T_l(x)w(x)dx = \begin{cases} \pi\delta_l^k, k = 0, \\ \dfrac{1}{2}\pi\delta_l^k, k \neq 0 \end{cases} \tag{1.11}$$

Examples of the first six polynomials are plotted at the 1.4

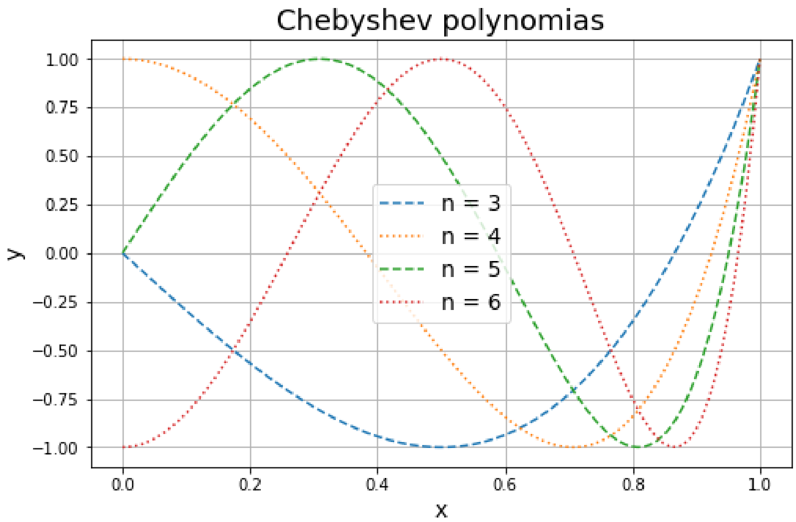


Figure 1.4: Chebyshev polynomias for n = 3, 4, 5, 6

These polynomials are used for the interpolation procedure to avoid the Runge phenomenon[2], in case, when they are monic polynomials. Moreover, the roots of them often used in numerical

[2]In the mathematical field of numerical analysis, Runge's phenomenon is a problem of oscillation at the edges of an interval that occurs when using polynomial interpolation with polynomials of high degree over a set of equispaced interpolation points.

linear algebra in method conjugated gradient descent, for example[3]. The details of the Chebyshev polynomials is not important for this work, but there are a lot of benefit of using them in different problems.

Chebyshev series is the expansion of the function by his polynomials or, simply substitute polynomials to (1.3):

$$f(x) = \sum_{k=0}^{\infty} c_k T_k(x)$$

$$c_k = \frac{1}{M}\int_{-1}^{1} f(x)T_k(x)w(x)dx, \text{where } M = \begin{cases} \pi, k = 0, \\ \frac{1}{2}\pi, k \neq 0 \end{cases}$$

$$\text{and } w(x) \text{ weight function } = \frac{1}{\sqrt{1-x^2}}$$

And the convergence theorem.

**Theorem 1.3** *When a function $f$ has $m+1$ continous derivatives on $[-1,1]$ or $f \in C^{m+1}[-1,1]$, where $m \in N^+$, then $\|f(x) - S_k(x)\| = \mathcal{O}\left(\frac{1}{k^m}\right)$ as $k \to \infty \quad \forall x \in [-1,1]$*

The proof here [19]. This theorem described the same fact that theorem 1.1 described for the Fourier expansion.

**Example of Chebyshev interpolation** Consider the function $y = sin(x) + xcos(x)$ and at the fig. 1.5 the results.

In fact, Chebyshev series is Generalized Fourier series:

$$f(x) = \sum_{i=1}^{N} c_n \phi_n(x)$$

$$\langle \phi_i, \phi_j \rangle = \int_V \phi_i \phi_j w dV = K\delta_i^j$$

### 1.2.3 Another functions and expansion over them

In case, when Generalized Fourier Series (GRS) is considered there are a lot of function can be used, for example[1]:

- Gegenbauer polynomials
- Jacobi polynomials

[3] More information is here - "E. Kaporin, Using Chebyshev polynomials and approximate inverse triangular factorizations for preconditioning the conjugate gradient method, Zh. Vychisl. Mat. Mat. Fiz., 2012, Volume 52, Number 2, 179–204"

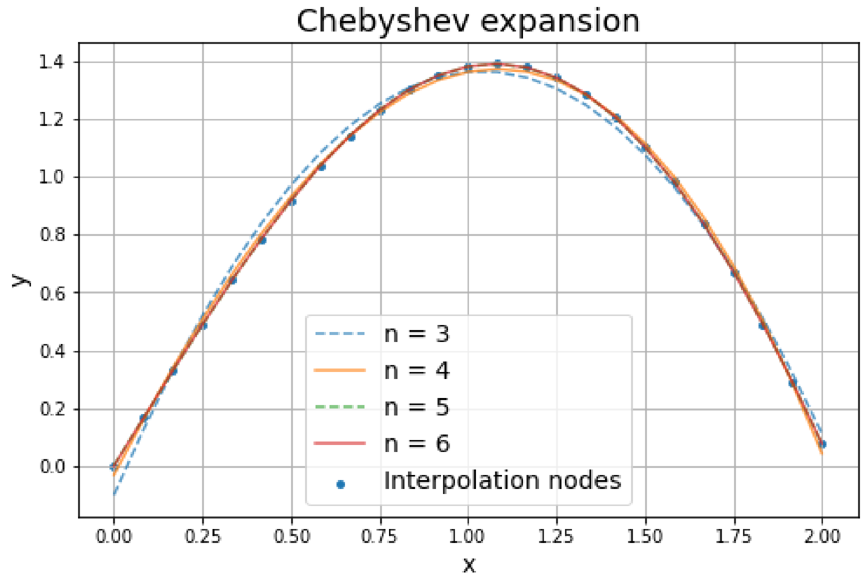


Figure 1.5: Chebyshev expansion with 3, 4, 5, 6 terms

- Romanovski polynomials
- Legendre polynomials

All have their own weight function and satisfy the GSR and potentially applicable for function approximation via the least-squares method and gradient-based methods with suitable penalty parameter (regularization term). In fact, during the approximation process (or supervised learning) includes fixation of the basis function, choice of the regularization. On the other hand, usage orthogonal functions for approximation or interpolation not restricted only GSR, actually, the linear regression model uses linearly-independent functions only.

**Function expansion over sigmoid function**

Let $\sigma = \dfrac{1}{1+e^{-x}}$ and substitute to (1.3) instead of $\phi_i$ and apply an affine transformation to argument:

$$G(x) = \sum_{i=1}^{K} \alpha_i \sigma(\beta_i x_i^j + \gamma_i) \tag{1.12}$$

the expansion over the sigmoid functions is got. This expansion, in fact, one of the widely used in approximation process. First of all, there is a useful theorem that provides guarantees of the quality for approximation.

**Theorem 1.4** *Let* $\sigma = \dfrac{1}{1+e^{-x}}$*, then finite sums of the form:*

$$G(x) = \sum_{i=1}^{K} \alpha_i \sigma(\beta_i x_i^j + \gamma_i)$$

*are dense in* $C(I_n{}^4)$*. In other words, given any* $f \in C(I_n)^5, \epsilon > 0$*, there is a sum,* $G(x)$*, of the above form, for which:* $\|G(x) - f(x)\| < \epsilon, \quad \forall x \in I_n$

In simple words, this theorem provides justification for the expansion of the function into the sigmoidal series, moreover, the quality of the approximation can be tremendously increased via increasing the terms in the series. By the way, the series (1.12) named neural network[6] or perceptron[7] with one hidden layer and sigmoidal activation function.

Coefficients for the expansion can be found via the least-squares method or using the gradient-based methods which more suitable when talking about neural networks. The question of estimating them is a future section question, but there is a lot of special algorithms for it.

[4] Unit cube in $R^n$. The term unit cube or unit hypercube is also used for hypercubes, or "cubes" in $n$-dimensional spaces, for values of n other than 3 and edge length 1

[5] The space of continuous functions over $I_n$

[6] Neural network, or Artificial neural networks (ANN) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules.

[7] In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers or regressor.

## 1.3 Solving differentials equations

### 1.3.1 Introduction

Differential equations can be split into two big groups:

- Ordinary differential equations (ODE)
  - Single ODE
  - System of ODEs
- Partial differential equations (PDE)
  - Single PDE
  - System of PDEs

For each group, there are a lot of solving methods, for ODE (shooting method, Euler method, Runge–Kutta methods, and so on) or for PDE (Finite difference method, finite element method, finite volume method and so on). Most of them based upon the idea of numerical integration or function approximation via some sequence of function and the minimization of the residual, for example, weighted residuals method [9] [10]. All of these methods lead to solving the system of algebraic equations, in the general case. In case when the equation is simple enough the system of linear equations should be solved. Suppose that after applying some numerical method for some problem and not important for what method and what problem, a linear system is got: $Ax = b$, let $b = \hat{b} + e_b$, where $e_b$ is error in vector $b$ it can be caused by rounding errors or predefined errors related to data collection if speech going about real problems of the oil and gas industry, for example. Here the existence of this error is interesting because the solution error implies from the error in the left-hand side and can be larger than her. $x = A^{-1}(\hat{b} + e) = A^{-1}\hat{b} + A^{-1}e_b$, and $x = \hat{x} + e_x = A^{-1}\hat{b} + A^{-1}e_b$ and:

$$\begin{cases} \hat{x} = A^{-1}\hat{b} \\ e_x = A^{-1}e_b \end{cases} \implies \max_{e,b} \frac{\|A^{-1}e_b\|}{\|A^{-1}\hat{b}\|}\frac{\|b\|}{\|e_b\|} = \|A\|\|A^{-1}\| = \kappa(A)$$

$\kappa(A)$ - condition number[11].

It means that if the matrix has a large value of the condition number then the error of the $x$ is large[8]. This fact leads to the use of preconditioners to decrease the condition number and gets a more

[8]The influence of the condition number on the solution accuracy presented at the fig. 1.6. Here considered the model example of the linear system, with $\kappa = 3422.83$ and presented the components of the vector b and his deviations, then x was found and deviations. It can be seen that the deviations of b (left part, blue circles) near the initial values(red line), but the deviations for x has a large spreading. This is an influence of condition number.

stable solution. There are a lot of ways to preconditioning the system of linear equations: Jacobi (or diagonal) preconditioner, incomplete Cholesky factorization, incomplete LU factorization, and so on.

It is one of the problems that arise during the ODE/PDE solving but in fact, there are problems such as convergence rate, mesh generation, interpolation of the solution, choose the function for approximation, and so on. In this work don't provide the method that ideal and works well for all problems, but for the presented in the continuation, problems work fast and well. In fact, if the method won't depend on the mesh and use only the randomly chosen points and will work well for some problems - it will be a small victory.

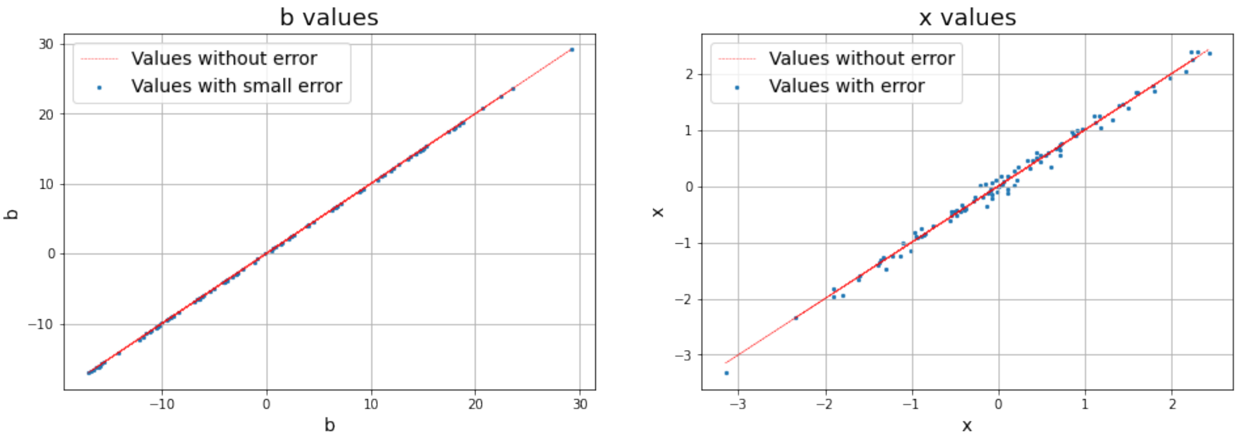


Figure 1.6: The influence of the condition number on the solution accuracy

### 1.3.2 Solving Ordinary differential equations

In this part consider the arbitrary ODE:

$$\begin{aligned} \mathcal{D}\left(x, y, y^{(1)}, \ldots, y^{(n)}\right) = 0, \quad x \in \Omega = [0,1] \subset R \\ B(y) = 0, \quad x \in \partial\Omega = \{0,1\} \end{aligned} \tag{1.13}$$

, where $\mathcal{D}(\dots)$ - differential operator, $B(y)$ - boundary conditions function.

$$B(y) = \begin{cases} D(y) = 0, & x \in \partial\Omega_D = \{0,1\} \\ N(y) = 0, & x \in \partial\Omega_N = \{0,1\} \end{cases} , \partial\Omega_N \cup \partial\Omega_D = \partial\Omega$$

where $D(y)$ - Dirichlet boundary conditions and $\partial\Omega_D$ boundary for these conditions, N(y) - Neumann boundary conditions and $\partial\Omega_N$ is bound them. All of the equations considered now and in the future in this work will be defined as (1.13).

For solving this equation will be considered two methods, weighted residuals method [10] (Bubnov-Galerkin) and finite differences method [6].

For concreteness, ODE example for consideration:

$$\frac{d}{dx^2}y(x) + 2y = sin(2x)\left(1 - 2sin(2x)\right), x \in [0, 1]$$
$$y(0) = 0, \quad \frac{d}{dx}y\Big|_{x=0} = 1$$

, where $y = \frac{1}{2}sin(2x)$ - analytical solution.

**Galerkin method**

The key idea is to define the solution as:

$$y_h = \phi_0 + \sum_{i=1}^{N} a_i\phi_i(x) \tag{1.14}$$

and $\phi_0$ satisfy all boundary conditions $\phi_0 : B(\phi_0) = 0 \implies \phi_0(0) = 0, \phi_0(1) = 1$ and $\phi_i$ satisfy the homogenous boundary conditions $\phi_i(0) = \phi_i(1) = 0$. The solution to the problem is a some weighted sum of linearly independent functions that satisfy the boundary conditions and in the general case satisfy the initial conditions too. For calculation, the coefficients use minimization residual method, where:

$$\min_{a_1,\ldots,a_n} R(a_1, \ldots, a_n) = \int_\Omega \mathcal{D}(y_h)d\Omega + \int_{\partial\Omega_N} N(y_h)d\partial\Omega$$

From [10] is known that $R$ does not equal zero in the general case and for evaluating the coefficients use the integration with weight function:

$$\int_\Omega wR(a_1, \ldots, a_n)d\Omega = \int_\Omega w\mathcal{D}(y_h)d\Omega + \int_{\partial\Omega_N} wN(y_h)d\partial\Omega = 0$$

, where $w = \psi_i$, weight functions. The residual and weight functions must be orthogonal. In a more general case without using $phi_0$ the residual is:

$$\int_\Omega \psi_j R(a_1, \ldots, a_n)d\Omega = \int_\Omega \psi_i \mathcal{D}\left(\sum_{i=1}^{N} a_i\phi_i(x)\right)d\Omega + \int_{\partial\Omega} \psi_j B\left(\sum_{i=1}^{N} a_i\phi_i(x)\right)d\partial\Omega =$$
$$= \int_\Omega \psi_j \mathcal{D}\left(\sum_{i=1}^{N} a_i\phi_i(x)\right)d\Omega + \int_{\partial\Omega} \psi_j B\left(\sum_{i=1}^{N} a_i\phi_i(x)\right)d\partial\Omega = \text{ if } \mathcal{D}, \mathcal{B} \text{ are linear operators } =$$
$$= \int_\Omega \sum_{i=1}^{N} a_i\psi_j\mathcal{D}(\phi_i(x))d\Omega + \int_{\partial\Omega} \sum_{i=1}^{N} a_i\psi_j B(\phi_i(x))d\partial\Omega = 0$$

From the equation above system of algebraic equations can be constructed and solve it for unknown coefficients $a_i$.

### Galerkin method, special case

If $w$ is delta Dirac function, then the Galerkin method also called the Pointwise collocation method, which more easy for implementation.

$$
\begin{gathered}
w = \delta(x - x_k), x_k \in X \subset \Omega, \text{ and } \|X\| = K, \\
\int_\Omega \delta(x - x_k) R(a_1, \ldots, a_n) d\Omega = \int_\Omega \delta(x - x_k) \mathcal{D}\left(\sum_{i=1}^N a_i \phi_i(x)\right) d\Omega + \\
+ \int_{\partial\Omega} \delta(x - x_k) B\left(\sum_{i=1}^N a_i \phi_i(x)\right) d\partial\Omega = \mathcal{D}\left(\sum_{i=1}^N a_i \phi_i(x_k)\right) + B\left(\sum_{i=1}^N a_i \phi_i(x_k)\right)
\end{gathered} \tag{1.15}
$$

In case when the number of points of collocation more then the number of unknown coefficients the problem solves via optimization techniques, the least-squares method for example.

### Finite difference method

First of all, the finite difference derivative is:

- Left derivative

$$\left.\frac{dy}{dx}\right|_{x=x_i} = \frac{y(x_i) - y(x_{i-1})}{x_i - x_{i-1}} \tag{1.16}$$

- Central derivative

$$\left.\frac{dy}{dx}\right|_{x=x_i} = \frac{y(x_{i+1}) - y(x_{i-1})}{x_{i+1} - x_{i-1}} \tag{1.17}$$

- Right derivative

$$\left.\frac{dy}{dx}\right|_{x=x_i} = \frac{y(x_{i+1}) - y(x_i)}{x_{i+1} - x_i} \tag{1.18}$$

Actually, the approximation quality better for the central difference derivative. The derivatives of higher order can be constructed from the first-order derivatives (left, right, central). For example:

$$\left.\frac{d^2y}{dx^2}\right|_{x=x_i} = \left.\frac{d}{dx}\right|_{x=x_i} \left[\frac{y(x_{i+1})}{x_{i+1} - x_i}\right] - \left.\frac{d}{dx}\right|_{x=x_i} \left[\frac{y(x_{i-1})}{x_i - x_{i-1}}\right] = \frac{y(x_{i+1}) - y(x_i)}{x_{i+1} - x_i} - \frac{y(x_i) - y(x_{i-1})}{x_i - x_{i-1}}$$

When the grid uniformly distributes the $x_i$ values: $x_{i+1} - x_i = d$:

$$\left.\frac{d^2y}{dx^2}\right|_{x=x_i} = \frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{d^2}$$

So, the idea of the FDM is to substitute the finite derivatives and solve algebraic equations.

$$\left.\mathcal{D}(x, y, y^{(1)}, \ldots, y^{(n)})\right|_{x=x_i} = \mathcal{D}\left(x_i, y(x_i), \frac{y(x_{i+1}) - y(x_{i-1})}{x_{i+1} - x_{i-1}}, \ldots, \frac{y(x_{i+n-1}) + \cdots + y(x_{i-n+1})}{d^n}\right)$$

And the same way for the boundary conditions:

$$B(y)\Big|_{x=x_i} = \begin{cases} D(y) = 0, & x \in \partial\Omega_D = \{0, 1\} \\ N(y) = 0, & x \in \partial\Omega_N = \{0, 1\} \end{cases}$$

After the solving equations, the values of $y_i$ are known and needed to be interpolated over the domain $\Omega$.

**Comparison of the provided methods**

Methods are very different, the FDM provides the solution in the fixed nodes and interpolates the solution from these nodes overall domain, on the other hand, the Galerkin method provides approximation solution in the mean sense over the domain. This difference makes the variability of the interpolation methods or basis functions for calibration of the numerical solution quality. The strong and ill sides of the FDM are high quality of the solution over the nodes, but the interpolation process leads to the Runge phenomenon, besides the size of the grid has a tremendous influence on the solution quality. The Galerkin method provides the approximation over the domain and strongly depends on the initial choice basis functions, so, there is the probability, that solution has a compact form.

It will be good if the strong sides of these methods will be combined into one approximator. Ideal case, when the number of terms increases, the solution quality increase too.

First of all, using the theorem 1.4 and the solution form (1.14):

$$y_h(x) = \phi_0(x) + \sum_{i=1}^{K} \alpha_i \sigma(\beta_i x + \gamma_i) \tag{1.19}$$

$\phi_0$ also satisfy the boundary conditions. For this solution from the theorem known, that the approximation quality strongly depends on the number of terms in the series, in addition, this form satisfies the boundary conditions, as in the Galerkin method. Now, the quality of the solution is guaranteed by the theorem and the question about basis function is solved. Moreover, using points collocation method:

$$\mathcal{L} = \frac{1}{|X|} \sum_{x \in X} \left[ \| R(x; p_1, \ldots, p_N) \|^2 \right], \quad X \in \Omega \subset R, p_i = (\alpha_i, \beta_i, \gamma_i) \in P \subset R^3$$

$$\textbf{Coefficients}: \min_{p_i} \mathcal{L} = \begin{cases} \dfrac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \beta_i} = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \gamma_i} = 0 \end{cases} \tag{1.20}$$

Currently, the solution is found using the least-squares method, which leads to solving the system of equations with not one solution. For each solution, the loss function should be calculated and chose the parameter where problem has a minimum value.

For this approach calculation of the derivatives for a differential operator should be provided:

$$\begin{aligned}\frac{dy_h}{dx} = \frac{d}{dx}\left[\phi_0(x) + \sum_{i=1}^{K} \alpha_i \sigma(\beta_i x + \gamma_i)\right] = \frac{d}{dx}\phi_0(x) + \sum_{i=1}^{K} \alpha_i \frac{d}{dx}\sigma(\beta_i x + \gamma_i) = \\ = \frac{d}{dx}\phi_0(x) + \sum_{i=1}^{K} \alpha_i \beta_i \sigma(\beta_i x + \gamma_i)(1 - \sigma(\beta_i x + \gamma_i))\end{aligned} \tag{1.21}$$

The form of the derivative immediately told that the solving of equations (1.21) is very unstable and there are a lot of roots. On the other hand, using the numerical derivative (1.16), (1.17), (1.18) leads to:

$$\left.\frac{dy_h}{dx}\right|_{x=x_i} \approx \frac{y_h(x_{i+1}) - y_h(x_{i-1})}{2d} = \frac{1}{2d}\left[y_h(x_{i+1}) - y_h(x_{i-1})\right] \tag{1.22}$$

**Artificial neural networks (ANN)**

Definition from Wikipedia is “Artificial neural networks (ANN) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems ”learn” to perform tasks by considering examples, generally without being programmed with task-specific rules“, or the second one definition: “A mathematical model, as well as its software or hardware implementation, built on the principle of organization and functioning of biological neural networks - networks of nerve cells of a living organism.”

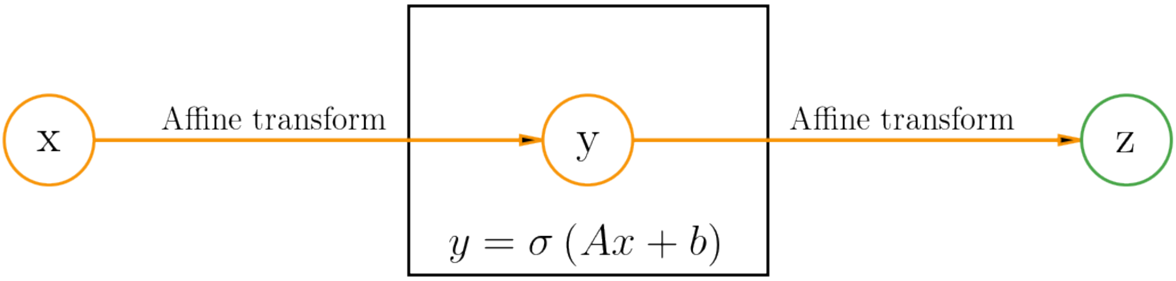


Figure 1.7: The illustration of (1.19). One layered neural network

These definitions are similar, in the sense that the input signal passes through the set of ordered simple operations or layers, and at the end of these operations the output is the result of the neural network. The order of these operations also called the architecture of the neural network. There are a lot of different types of layers[9], the most widely used is the fully connected layer or

[9]The zoo of neural network types: ANN zoo

dense layer as in figure 1.7. Looking more precisely the neural network is sequence of affine transformations (edges) and nonlinear transformation (nodes):

$$\mathcal{N}(x) = \left[A^2 \circ \phi^1 \circ A^1\right](x) = A^2\phi^1\left(A^1x + b^1\right) + b^2$$
$$A^1 \in R^{m\times n}, A^2 \in R^{k\times m}, b^1 \in R^m, b^2 \in R^k, x \in R^n$$

In general case $l$ layered neural network is:

$$\mathcal{N} = A^l \circ \phi^{l-1} \circ A^{l-1} \circ \dots \circ \phi^1 \circ A^1 = A^l\left[\phi^{l-1}\left[\dots\left[A^1(x) + b^1\right]\dots\right] + b^{l-1}\right] + b^l \tag{1.23}$$

where $A^i, \forall i \in \{1, \dots, l\}$ is the parameter that must be found. For the successful using the neural networks:

- Define the architecture
- Define the loss function
- Choose a suitable optimization algorithm
  - How the optimization process looks
  - Existing optimization algorithms

**Optimization part. Backpropagation algorithm**

The main goal is getting the numerical solution of the DE and for this aim is to use the residual (1.20) and minimize it over the parameters of the neural network:

$$\min_{A^l,b^l,\dots,A^1,b^1} \mathcal{L} = \min_{A^l,b^l,\dots,A^1,b^1} \mathcal{L} = \frac{1}{|X|}\sum_{x\in X} \|R(x)\|^2$$

Now, how to minimize this complex function? Using the least-squares leads to solving the equations or use gradient-based optimization. For the estimation, the values of the neural network parameters use the gradient-based methods and iteratively goes to the local minimum(!). Suppose, the for the point collocation method randomly choose the set of points at the $k$-th step, the loss is calculated and gradients are calculated:.

$$\nabla A_k^l = \nabla_{A^l}\mathcal{L}_k, \quad A_{k+1}^l = A_k^l - \lambda(k)\psi(\nabla A_k^l) \tag{1.24}$$

the $\psi$ is the main part of the particular algorithm because using the $\psi(x) = x$, stochastic gradient descent (SGD) immediately have gotten. Using different $\psi$, the corresponding methods are obtained [24], [8], [14], [7].

**Optimizers comparison** To demonstrate the quality of various optimization algorithms, a simple neural network architecture was chosen and trained to approximate the function. Lines are the average value of the loss function at a particular iteration, the region of the corresponding color is the region in which the error may lie on average. To collect such statistics, the neural network was trained by each optimizer 25 times. The results are presented in the figure 1.8.

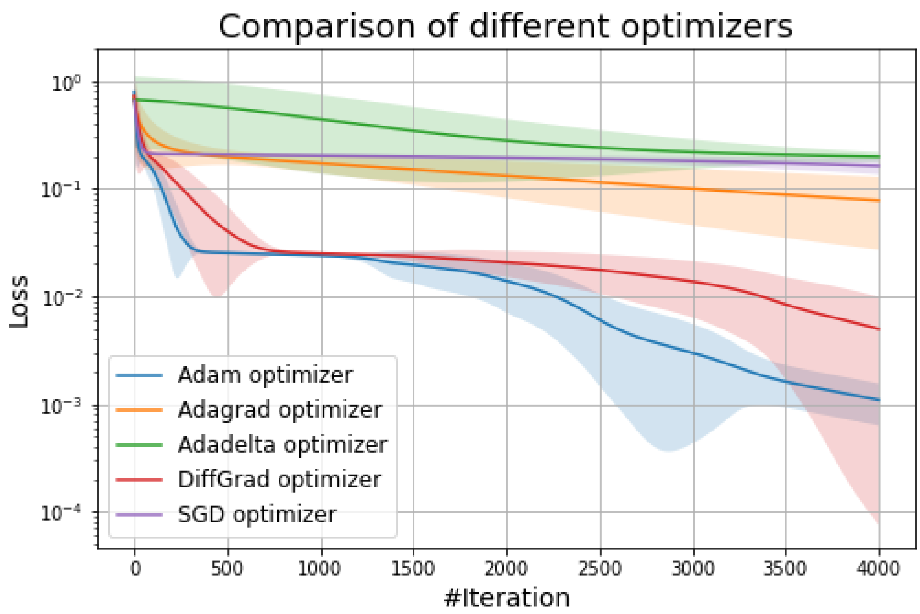


Figure 1.8: Comparison of different optimizers for fixed neural network architecture

Consider the sequence of the operators (1.23) and the quality function or loss function $\mathcal{L}$. Currently not important what loss function and the nature of the operators:

$$\mathcal{N} = A^l \circ \phi^{l-1} \circ A^{l-1} \circ \cdots \circ \phi^1 \circ A^1, \quad \mathcal{L} = \mathcal{L}(\mathcal{N})$$

For efficient evaluating the gradients over the parameters exists a backpropagation[10] algorithm [5].

[10] In machine learning, backpropagation (backprop, BP) is a widely used algorithm in training feedforward neural networks for supervised learning. Generalizations of backpropagation exist for other artificial neural networks (ANNs), and for functions generally – a class of algorithms referred to generically as ”backpropagation” - from Wikipedia

The key idea is to use the chain rule for the derivative:

$$
\begin{cases}
\dfrac{\partial \mathcal{L}}{\partial A^l} = \nabla_{A^l} \mathcal{L} \\
\dfrac{\partial \mathcal{L}}{\partial A^{l-1}} = \left[\phi^l\right]' \left[A^{l-1}\right]^T \nabla_{A^l} \mathcal{L} \\
\dfrac{\partial \mathcal{L}}{\partial A^{l-2}} = \left[\phi^{l-1}\right]' \left[A^{l-2}\right]^T \left[\phi^l\right]' \left[A^{l-1}\right]^T \nabla_{A^l} \mathcal{L} = \left[\phi^{l-1}\right]' \left[A^{l-2}\right]^T \dfrac{\partial \mathcal{L}}{\partial A^{l-1}} \\
\text{For k-th derivative in the same way:} \\
\dfrac{\partial \mathcal{L}}{\partial A^{l-k}} = \left[\phi^{l-k+1}\right]' \left[A^{l-k}\right]^T \dfrac{\partial \mathcal{L}}{\partial A^{l-k+1}}
\end{cases}
$$

Now it is known how a neural network works, how it is trained and why a solution can be built with arbitrary accuracy. Next, we will consider different architectures of neural networks for solving different problems, and different approaches, for example, the approach based on the Galerkin method, when the Dirichlet boundary conditions are embedded in a neural network. There is also an approach based on the Ritz method that reduces the solution of the equation to an extremal problem. For example, when solving equations, it is possible to integrate the boundary conditions into the approximator structure [16] [18]. Here the solution is presented in the form:

$$y_h = A(x) + B(x)\mathcal{N}(x) \tag{1.25}$$

where $A$ satisfies the boundary conditions of the first and second kind, where $A$ satisfies the boundary conditions of the first and second kind, and $B$ is in a sense a function of distance, or rather a function that “removes” the values of the model (neural network) at the boundary.

**Example** Consider equation $\phi\left(x, y, \dfrac{dy}{dx}, \dfrac{d^2y}{dx^2}\right) = 0$ and boundary conditions $y(0) = y_0, y(1) = y_1$. In this case, the solution will be built in the form:

$$y_h = (1-x)y_0 + xy_1 + (1-x)x\mathcal{N}(x)$$

Thus, for such a form, a neural network is only part of the solution, for points within a region. It is clear that the name of the complex boundary conditions for the partial differential equation to construct a solution in this form is very difficult. In this form, it is convenient to search for a solution having homogeneous boundary conditions of the first kind. You can use the results from [10], where it is proposed to construct the solution in such a way as to satisfy only conditions of the first kind, and transfer conditions of the second and third kind to the neural network again (to the loss function), example:

$$\phi\left(x, y, \frac{dy}{dx}, \frac{d^2y}{dx^2}\right) = 0, \quad y(0) = y_0, \frac{dy}{dx}\Big|_{x=0} = y_1$$
$$y_h = (1-x)y_0 + B(x)\mathcal{N}(x), \quad \mathcal{L}^{'} = \mathcal{L} + \lambda \left\| \frac{dy_h}{dx}\Big|_{x=0} - y_1 \right\|$$

Another approach [4] also embeds the boundary conditions in the general solution, however, it occurs due to an additional term that estimates the error between the conditions and the solution itself at the boundary and embeds the additional term in a row in order to satisfy the conditions. In fact, every few iterations of the network training, the term is recalculated (a small system of equations is solved) and adjusted to the boundary conditions. Not quite an easy way to implement, however, the quality of the final solution depends on the boundary conditions, on the structure of the additional unit and the necessary accuracy. Models based on the Galerkin method are quite common, so the authors [22] proposed the structure of the model so that, with an increase in the dimension of the problem, the quality of the solution remains acceptable. Their model looks interesting, combines many breakthrough deep learning approaches, but in view of this, the speed of learning is very low. The authors themselves in their work provide an assessment of the training time and the necessary capacities for this - it takes an order of magnitude more time on a conventional personal computer than classical approaches require, but the main goal is high-dimensional tasks, where the algorithm really showed good quality. All approaches proposed and considered below can be divided into 2 groups:

- Embed in the solution itself [16] [18] [4]
- Consider a conditional problem solved by the Lagrange method. In the learning process, the model learns not only to solve the equation itself, but is also fined for not satisfying the boundary conditions [21]

Each group has its own characteristics, so for methods from the first group, the high quality of the solution is characteristic, but the difficulty of drawing up the presentation of the solution is high. The second group is characterized by a not very high quality solution, especially at the borders, however, with sufficient training time and properly selected regularization, this problem is solved, but the plus is the ease of implementation.

**Examples of ODE**

$$\frac{dy}{dx} = sin(x), \quad y(0) = -1$$

| Method | Parameters num | Accuracy |
|---|---|---|
| FDM | 10 | $7.8210^{-3}$ |
| FDM | 25 | $2.8810^{-3}$ |
| FDM | 50 | $1.4410^{-3}$ |
| FDM | 100 | $0.6910^{-3}$ |
| FDM | 200 | $0.3410^{-3}$ |
| ANN | 8 | $0.29810^{-3}$ |
| ANN | 10 | $0.11110^{-3}$ |
| ANN | 20 | $0.010510^{-3}$ |
| ANN | 50 | $0.0041310^{-3}$ |

# Bibliography


[1] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. Applied mathematics series. Dover Publications, 1965.

[2] Maxime Bocher. *Introduction to the Theory of Fourier's Series*. Mathematics Department, Princeton University, Annals of Mathematics, Second Series, Vol. 7, No. 3, 2020.

[3] Terence Candes, Emmanuel; Tao. The dantzig selector: Statistical estimation when p is much larger than n. pages 2313—2351, 2007.

[4] Linlin Cao, Ran He, and Bao-Gang Hu. Locally imposing function for generalized constraint neural networks - a study on equality constraints, 2016.

[5] Y. Chauvin and D.E. Rumelhart. *Backpropagation: Theory, Architectures, and Applications*. Developments in Connectionist Theory Series. Taylor & Francis, 2013.

[6] I. Dimov, I. Faragó, and L. Vulkov. *Finite Difference Methods. Theory and Applications: 7th International Conference, FDM 2018, Lozenetz, Bulgaria, June 11-16, 2018, Revised Selected Papers*. Lecture Notes in Computer Science. Springer International Publishing, 2019.

[7] Shiv Ram Dubey, Soumendu Chakraborty, Swalpa Kumar Roy, Snehasis Mukherjee, Satish Kumar Singh, and Bidyut Baran Chaudhuri. diffgrad: An optimization method for convolutional neural networks, 2019.

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[9] B.A. Finlayson. *The Method of Weighted Residuals and Variational Principles*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2013.

[10] C.A.J. Fletcher. *Computational Galerkin Methods*. Scientific Computation. Springer Berlin Heidelberg, 2012.

[11] J.E. Gentle. *Matrix Algebra: Theory, Computations, and Applications in Statistics*. Springer Texts in Statistics. Springer, 2007.

[12] S.S. Haykin. *Neural Networks: A Comprehensive Foundation*. International edition. Prentice Hall, 1999.

[13] Arthur E. Hoerl; Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. pages 55—67, 1970.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[15] R. Kress. *Numerical Analysis*. Graduate Texts in Mathematics. Springer New York, 2012.

[16] I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[17] Ji Zhu Li Wang, Michael D. Gordon. Regularized least absolute deviations regression and an efficient algorithm for parameter tuning. pages 690—700, 2006.

[18] Zeyu Liu, Yantao Yang, and Qing-Dong Cai. Solving differential equation with constrained multilayer feedforward network, 2019.

[19] J.C. Mason and D.C. Handscomb. *Chebyshev Polynomials*. CRC Press, 2002.

[20] Weijie Su Emmanuel J. Candes Małgorzata Bogdan, Ewout van den Berg. Statistical estimation and testing via the ordered l1 norm. 2013.

[21] G. P. Purja Pun, R. Batra, R. Ramprasad, and Y. Mishin. Physically informed artificial neural networks for atomistic modeling of materials. *Nature Communications*, 10(1), May 2019.

[22] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, Dec 2018.

[23] Robert Tibshirani. Regression shrinkage and selection via the lasso. pages 267—288, 1996.

[24] Matthew D. Zeiler. Adadelta: An adaptive learning rate method, 2012.