Physics Informed Neural Network using Isogeometric Analysis

Maciej Paszyński

AGH University of Science and Technology, Kraków, Poland
maciej.paszynski@agh.edu.pl
home.agh.edu.pl/paszynsk

1 Introduction

Isogeometric Analysis (IGA) [1] employes smooth high-order and continuity base functions for approximation of solutions of Partial Differential Equations (PDEs). Physics Informed Neural Networks (PINN) [2] approximates the solution of a given PDEs with Deep Neural Network (DNN) being the concatenation of several linear operators and non-linear activation functions. The Stochasstic Gradient Descent (SGD) [3] is used to find the coefficients of the DNN approximating a given PDEs. In [4], we described how IGA can be used to approximate the coefficients of linear combination of B-splines, employed for solution of a family of PDEs depending on the right-hand side and boundary condition functions. In this work we focus on incorporation of PINN and IGA. We focus our attention on simple one-dimensional PDE.

Following [5], let us introduce the knot vector [0 0 0 1 1 1] defining the quadratic B-spline basis functions with C^0 separators

$$B_{1,2}(x) = (1-x)^2; \quad B_{2,2}(x) = 2x(1-x); \quad B_{3,2}(x) = x^2 \tag{1}$$

Let us introduce the problem

$$-u''(x) = f_n(x) \quad x \in (0, 0.5) \tag{2}$$

defined over $x \in (0, 0.5)$, with boundary conditions $u(0) = 0$ and $u'(0.5) = g(x)$. We setup $g(x) = n\pi cos(n\pi x)$ and $f(x) = n^2\pi^2 sin(n\pi x)$. The family of solution of this problem are

$$f_n(x) = sin(n\pi x) \tag{3}$$

2 Physics Informed Neural Network

2.1 Formulation

We define the neural network

$$PINN(x) = u \tag{4}$$

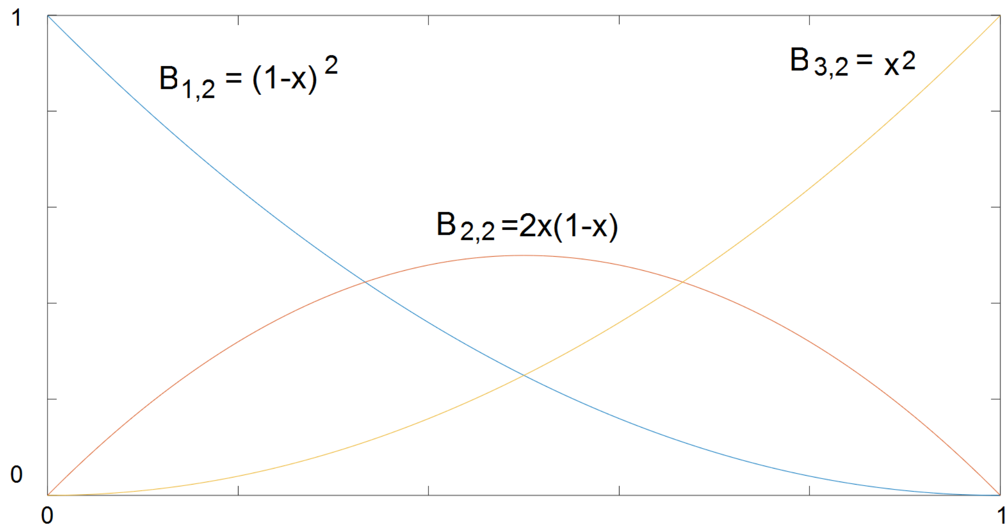

Fig. 1: Three B-splines over a single interval (element)

where

$$PINN(x) = c\sigma\left(ax+b\right) + d = \frac{c}{1+exp(-ax-b)} + d \tag{5}$$

We compute the derivatives

$$PINN_x(x) = \frac{a*c*exp(-ax-b)}{\left(exp(-ax-b)+1\right)^2} \tag{6}$$

and

$$PINN_{xx}(x) = c\left(\frac{2a^2exp(-2ax-2b)}{\left(exp(-ax-b)+1\right)^3} - \frac{a^2exp(-ax-b)}{\left(exp(-ax-b)+1\right)^2}\right) \tag{7}$$

2.2 Training

The goal of the training is to find values of the weights a, b, c, d

We prepare a set of samples

- We randomly select $x \in (0, 0.5)$
- Input data x, output data $u = PINN(x)$

We define

$$F(x) = PINN_{xx}(x) + n^2\pi^2 sin(n\pi x) \tag{8}$$

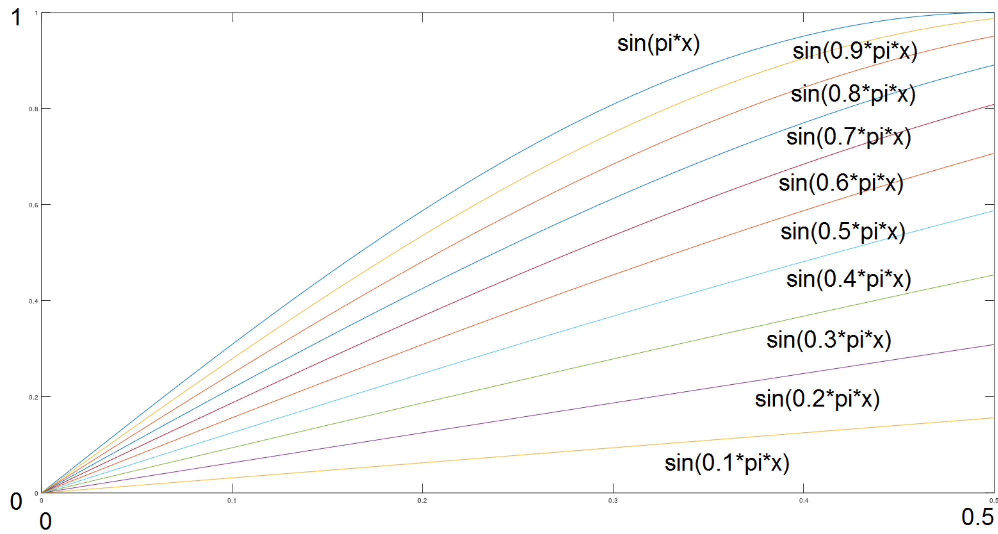

Fig. 2: Plot of different solutions $f_n(x)$ for $n = 0.1, 0.2, ..., 1$.

All the derivatives of PINN can be computed using Wolphram Alpha. We define the error of approximation of PDE

$$error1(x) = 0.5 * F(x)^2 = 0.5 * \left(PINN_{xx}(x) + n^2\pi^2 sin(n\pi x)\right)^2 =$$
$$0.5 * \left(c \left(\frac{2a^2 exp(-2ax - 2b)}{(exp(-ax - b) + 1)^3} - \frac{a^2 exp(-ax - b)}{(exp(-ax - b) + 1)^2} \right) + n^2\pi^2 sin(n\pi x) \right)^2 \quad (9)$$

as well as the error of approximation of the boundary condition at $x = 0$

$$error2(0) = 0.5 * (PINN(0) - 0)^2 = 0.5 * \left(\frac{c}{1 + exp(-b)} + d - 0 \right)^2 \quad (10)$$

as well as the error of approximation of the boundary condition at $x = 0.5$

$$error3(0.5) = 0.5 * (PINN_x(0.5) - g(0.5))^2 =$$
$$0.5 * \left(\frac{a * c * exp(-a0.5 - b)}{(exp(-a0.5 - b) + 1)^2} - n\pi cos(n\pi 0.5) \right)^2 \quad (11)$$

1. Select x
2. Compute $u = PINN(x) = c\sigma\left(ax + b\right) + d = \frac{c}{1+exp(-ax-b)} + d$
3. Compute $error1(x)$, $error2(0)$, $error3(0.5)$
4. Compute $\frac{\partial error1(x)}{\partial a}, \frac{\partial error1(x)}{\partial b}, \frac{\partial error1(x)}{\partial c}, \frac{\partial error1(x)}{\partial d}$
5. Compute $\frac{\partial error2(0)}{\partial a}, \frac{\partial error2(0)}{\partial b}, \frac{\partial error2(0)}{\partial c}, \frac{\partial error2(0)}{\partial d}$
6. Compute $\frac{\partial error3(0.5)}{\partial a}, \frac{\partial error3(0.5)}{\partial b}, \frac{\partial error3(0.5)}{\partial c}, \frac{\partial error3(0.5)}{\partial d}$

7. Correct

$$a = a - \eta * \frac{\partial e(x)}{\partial a} \tag{12}$$
$$b = b - \eta * \frac{\partial e(x)}{\partial b} \tag{13}$$
$$c = c - \eta * \frac{\partial e(x)}{\partial c} \tag{14}$$
$$d = d - \eta * \frac{\partial e(x)}{\partial d} \tag{15}$$

where $e(x) = error1(x) + error2(x) + error3(x)$

for $\eta \in (0, 1)$.

We compute

$$\frac{\partial PINN_{xx}(x)}{\partial a} = c\left(\frac{a^2 x * exp(-ax-b)}{(exp(-ax-b)+1)^2} - \frac{6a^2 x * exp(-2ax-2b)}{(exp(-ax-b)+1)^3} + \frac{6a^2 x * exp(-3ax-3b)}{(exp(-ax-b)+1)^4} - \frac{2aexp(-ax-b)}{(exp(-ax-b)+1)^2} + \frac{4a * exp(-2ax-2b)}{(exp(-ax-b)+1)^3}\right) \tag{16}$$

$$\frac{\partial PINN_{xx}(x)}{\partial b} = c\left(\frac{a^2 exp(ax+b)\left(-4exp(ax+b) + exp(2ax+2b) + 1\right)}{(exp(ax+b)+1)^4}\right) \tag{17}$$

$$\frac{\partial PINN_{xx}(x)}{\partial c} = \left(\frac{2a^2 exp(-2ax-2b)}{\left(exp(-ax-b)+1\right)^3} - \frac{a^2 exp(-ax-b)}{\left(exp(-ax-b)+1\right)^2}\right) \tag{18}$$

$$\frac{\partial PINN_{xx}(x)}{\partial d} = 0 \tag{19}$$

$$\frac{\partial PINN(0)}{\partial a} = 0 \tag{20}$$

$$\frac{\partial PINN(0)}{\partial b} = \frac{exp(-b)c}{\left(exp(-b)+1\right)^2} \tag{21}$$

$$\frac{\partial PINN(0)}{\partial c} = \frac{1}{(exp(-b)+1)} \tag{22}$$

$$\frac{\partial PINN(0)}{\partial d} = 1.0 \tag{23}$$

$$\frac{\partial PINN_x(0.5)}{\partial a} = c * exp(b-a)\frac{((1-0.5a) * exp(2a+b) + (0.5a+1)exp(1.5a))}{\left(exp(0.5a+b)+1\right)^3} \tag{24}$$

$$\frac{\partial PINN_x(0.5)}{\partial b} = \frac{ac * exp(b-0.5a)\left(exp(a) - exp(1.5a+b)\right)}{\left(exp(0.5a+b)+1\right)^3} \tag{25}$$

$$\frac{\partial PINN_x(0.5)}{\partial c} = \frac{a * exp(-0.5a - b)}{(exp(-0.5a - b) + 1)^2} \tag{26}$$

$$\frac{\partial PINN_x(0.5)}{\partial d} = 0 \tag{27}$$

Using the above formulas we have

$$\frac{\partial error1(x)}{\partial a} = \left(c\left(\frac{2a^2exp(-2ax-2b)}{(exp(-ax-b)+1)^3} - \frac{a^2exp(-ax-b)}{(exp(-ax-b)+1)^2}\right) + n^2\pi^2 sin(n\pi x)\right)$$
$$c\left(\frac{a^2x * exp(-ax-b)}{(exp(-ax-b)+1)^2} - \frac{6a^2x * exp(-2ax-2b)}{(exp(-ax-b)+1)^3}\right.$$
$$\left.+\frac{6a^2x * exp(-3ax-3b)}{(exp(-ax-b)+1)^4} - \frac{2aexp(-ax-b)}{(exp(-ax-b)+1)^2} + \frac{4a * exp(-2ax-2b)}{(exp(-ax-b)+1)^3}\right) \tag{28}$$

$$\frac{\partial error1(x)}{\partial b} = \left(c\left(\frac{2a^2exp(-2ax-2b)}{(exp(-ax-b)+1)^3} - \frac{a^2exp(-ax-b)}{(exp(-ax-b)+1)^2}\right) + n^2\pi^2 sin(n\pi x)\right)$$
$$c\left(\frac{a^2exp(ax+b)\left(-4exp(ax+b) + exp(2ax+2b) + 1\right)}{(exp(ax+b)+1)^4}\right) \tag{29}$$

$$\frac{\partial error1(x)}{\partial c} = \left(c\left(\frac{2a^2exp(-2ax-2b)}{(exp(-ax-b)+1)^3} - \frac{a^2exp(-ax-b)}{(exp(-ax-b)+1)^2}\right) + n^2\pi^2 sin(n\pi x)\right)$$
$$\left(\frac{2a^2exp(-2ax-2b)}{(exp(-ax-b)+1)^3} - \frac{a^2exp(-ax-b)}{(exp(-ax-b)+1)^2}\right) \tag{30}$$

$$\frac{\partial error1(x)}{\partial d} = 0 \tag{31}$$

$$\frac{\partial error2(x)}{\partial a} = 0 \tag{32}$$

$$\frac{\partial error2(x)}{\partial b} = \left(\frac{c}{1 + exp(-b)} + d - 0\right)\frac{exp(-b)c}{(exp(-b)+1)^2} \tag{33}$$

$$\frac{\partial error2(x)}{\partial c} = \left(\frac{c}{1 + exp(-b)} + d - 0\right)$$
$$\left(\frac{2a^2exp(-2ax-2b)}{(exp(-ax-b)+1)^3} - \frac{a^2exp(-ax-b)}{(exp(-ax-b)+1)^2}\right) \tag{34}$$

$$\frac{\partial error2(x)}{\partial d} = \left(\frac{c}{1+exp(-b)} + d - 0\right) \tag{35}$$

$$\frac{\partial error3(x)}{\partial a} = \left(\frac{a*c*exp(-a0.5-b)}{(exp(-a0.5-b)+1)^2} - n\pi cos(n\pi 0.5)\right)$$
$$c*exp(b-a)\frac{((1-0.5a)*exp(2a+b)+(0.5a+1)exp(1.5a))}{(exp(0.5a+b)+1)^3} \tag{36}$$

$$\frac{\partial error3(x)}{\partial b} = \left(\frac{a*c*exp(-a0.5-b)}{(exp(-a0.5-b)+1)^2} - n\pi cos(n\pi 0.5)\right)$$
$$\frac{ac*exp(b-0.5a)\,(exp(a)-exp(1.5a+b))}{(exp(0.5a+b)+1)^3} \tag{37}$$

$$\frac{\partial error3(x)}{\partial c} = \left(\frac{a*c*exp(-a0.5-b)}{(exp(-a0.5-b)+1)^2} - n\pi cos(n\pi 0.5)\right)$$
$$\frac{a*exp(-0.5a-b)}{(exp(-0.5a-b)+1)^2} \tag{38}$$

$$\frac{\partial error3(x)}{\partial d} = 0 \tag{39}$$

2.3 MATLAB implementation

```
% Creation of dataset
i=1;
n=0.333;
for x=0.01:0.01:0.5
y=sin(n*pi*x);
dataset_in_x(i)=x;
dataset_y(i)=y;
i=i+1;
endfor
ndataset=i-1;

  % Training
a1=1.0; a2=1.0; b=1.0; c=3.0; d=1.0;
eta=0.1;
r = 0 + (1-0).*rand(ndataset,1);
r=r.*ndataset;
for j=1:ndataset
```

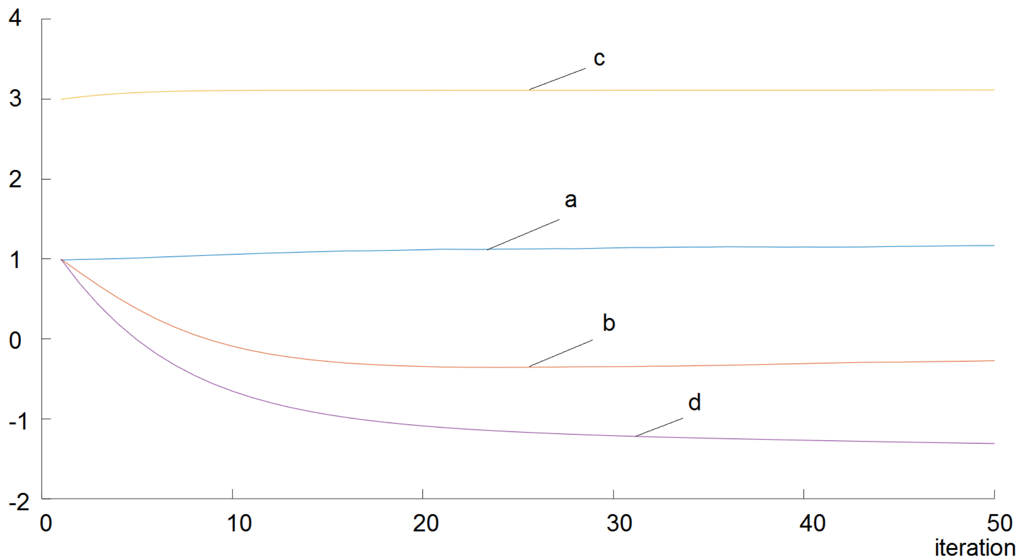

Fig. 3: Training of the simple PINN (5) starting from $a = b = d = 1.0$, and $c = 3.0$, for $\eta = 0.1$.

```
i=floor(r(j));
eval = c*1.0/(1.0+exp(-(a*dataset_in_x(i)+b)))+d;
error = 0.5*(eval-dataset_y(i))^2;
% Training of the PDE
x = dataset_in_x(i);
n = dataset_in_n(i);
Fx = c*(2*a*a*exp(-2*a*x-2*b) / power((exp(-a*x-b)+1),3)
- a*a*exp(-a*x-b) / power((exp(-a*x-b)+1),2))+n*n*pi*pi*sin(n*x);
derror1da = Fx*c *( (a*a*x*exp(-a*x-b))/power((exp(-a*x-b)+1),2)
-(6*a*a*x*exp(-2*a*x-2*b))/power((exp(-a*x-b)+1),3)
+(6*a*a*x*exp(-3*a*x-3*b))/power((exp(-a*x-b)+1),4)
-(2*a*exp(-a*x-b))/power((exp(-a*x-b)+1),2)
+(4*a*exp(-2*a*x-2*b))/power((exp(-a*x-b)+1),3) );
a=a-eta* derror1da;
derror1db = Fx*c*( (a*a*exp(a*x+b))*(-4*exp(a*x+b)
+exp(2*a*x+2*b)+1)/power((exp(a*x+b)+1),4));
b=b-eta* derror1db;
derror1dc = Fx*( (2*a*a*exp(-2*a*x-2*b))/power((exp(-a*x-b)+1),3)
-(a*a*exp(-a*x-b))/power((exp(-a*x-b)+1),2));
c=c-eta* derror1dc;
derror1dd = 0;
d=d-eta* derror1dd;
% Training of the boundary condition at x=0
```

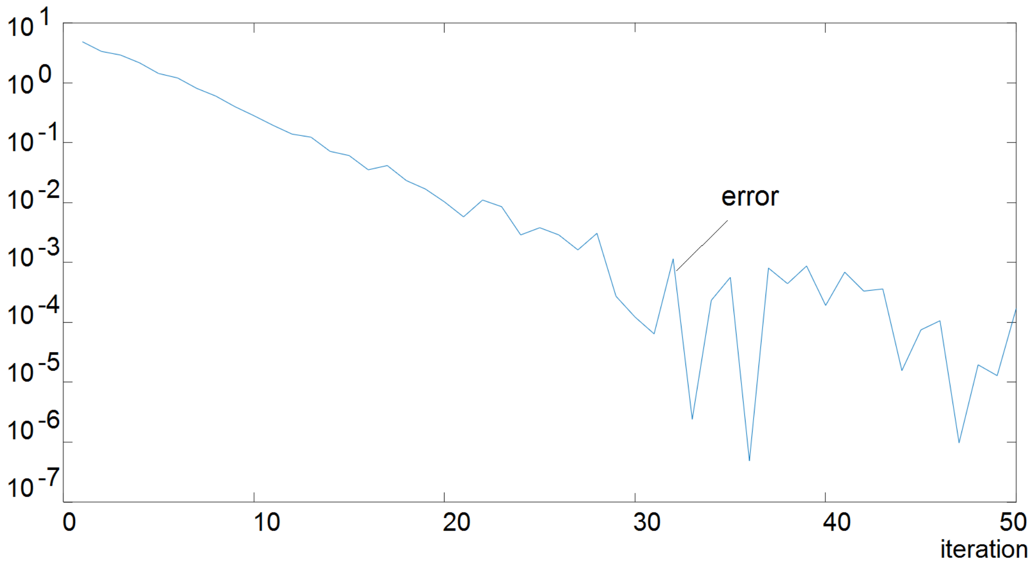

Fig. 4: Convergence of error for training of PINN

```
x=0;
derror2da = 0;
a=a-eta* derror2da;
derror2db = (c / (1+exp(-b))+d)* (exp(-b)*c)/power((exp(-b)+1),2);
b=b-eta* derror2db;
derror2dc = (c/(1+exp(-b))+d)*( (2*a*a*exp(-2*a*x-2*b))/power(exp(-a*x-b)+1,3)
- (a*a*exp(-a*x-b))/power(exp(-a*x-b)+1,2) );
c=c-eta* derror2dc;
derror2dd = c/(1+exp(-b))+d;
d=d-eta* derror2dd;
% Training of the boundary condition at x=0.5
x=0.5;
derror3da = (a*c*exp(-a*0.5-b)/power((exp(-a*0.5-b)+1),2)
-n*pi*cos(n*pi*0.5)) *c*exp(b-a)*((1-0.5*a)*exp(2*a+b)
+(0.5*a+1)*exp(1.5*a))/power((exp(0.5*a+b)+1),3);
a=a-eta* derror3da;
derror3db = (a*c*exp(-a*0.5-b)/power((exp(-a*0.5-b)+1),2)
-n*pi*cos(n*pi*0.5))* a*c*exp(b-0.5*a)*(exp(a)+exp(1.5*a+b))/power((exp(0.5*a+b)+1),3);
b=b-eta* derror3db;
derror3dc = (a*c*exp(-a*0.5-b)/power((exp(-a*0.5-b)+1),2)
-n*pi*cos(n*pi*0.5))* a*exp(-b-0.5*a)/power((exp(-0.5*a-b)+1),2);
c=c-eta* derror3dc;
derror3dd = 0;
d=d-eta* derror3dd;
```

```
endfor

   % evaluation of PINN approximation of sin(0.333*pi*x)
n=0.333;
x=0:0.01:0.5;
y=sin(n*pi.*x);
eval = c*1.0./(1.0+exp(-(a.*x+b)))+d;
plot(x,y,x,eval);
```

2.4 Verification

In Figure 3 we present the training over 50 samples, and in Figure 4 we present the convergence of the training.

The PINN has been trained for $n = 0.333$ so we compute

$$y(x) = ANN(n,x) = \frac{c}{1 + exp(a * x - b)} + d \tag{40}$$

we compare with $sin(0.333\pi x)$ in Figure 5.

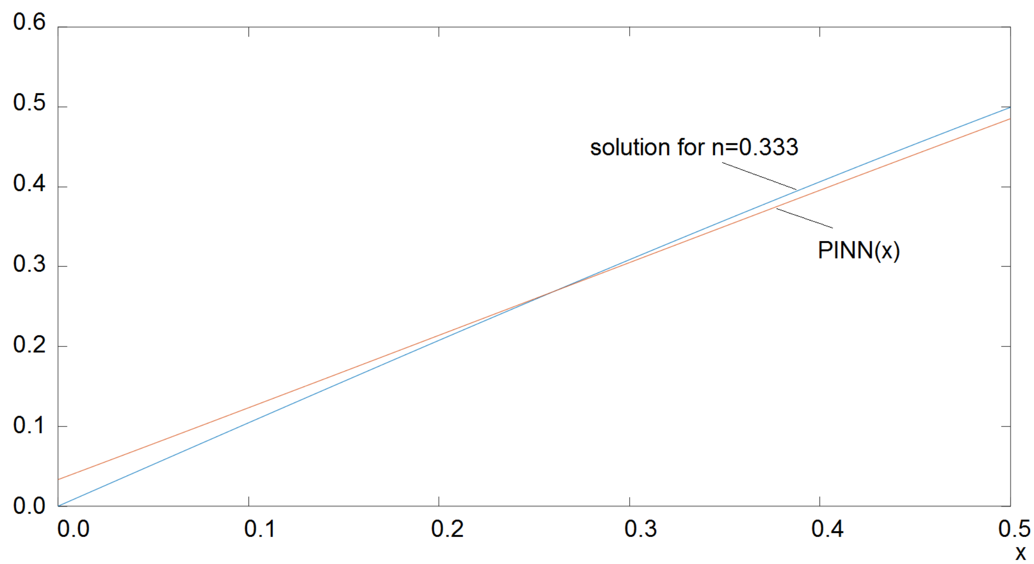

Fig. 5: Verification of the PINN trained for $n = 0.333$ with $y(x) = PINN(n,x) = \frac{c}{1+exp(a*x-b)} + d$

3 Physics Informed Neural Network for Isogeometric Analysis

The goal of this work is to investigate an alternative approach. We will approximate the solution of PDE with linear combination of B-spline base functions of order p, assuming continuity C^{p-1}

$$y = u(x) = \sum_i u_i B_{i,p}(x), \quad u'(x) = \sum_i u_i B'_{i,p}(x), \quad u''(x) = \sum_i u_i B''_{i,p}(x) \tag{41}$$

We define the loss and error functions for the approximation of PDE

$$\mathcal{L}_1(x) = \left(\sum_i u_i B''_{i,p}(x) - f(x) \right), \quad error_1 = \frac{1}{2} \left(\mathcal{L}_1(x) \right)^2 = \frac{1}{2} \left(\sum_i u_i B''_{i,p}(x) - f(x) \right)^2, \tag{42}$$

Similarly, we define the loss and error function for approximation of the Neumann

$$\mathcal{L}_2(x) = \left(\sum_i u_i B'_{i,p}(0) - g \right), \quad error_2 = \frac{1}{2} \left(\mathcal{L}_2(x) \right)^2 = \frac{1}{2} \left(\sum_i u_i B'_{i,p}(0) - g \right)^2, \tag{43}$$

and Dirichlet boundary conditions

$$\mathcal{L}_3(x) = \left(\sum_i u_i B_{i,p}(1) \right), \quad error_3 = \frac{1}{2} \left(\mathcal{L}_3(x) \right)^2 = \frac{1}{2} \left(\sum_i u_i B_{i,p}(1) \right)^2. \tag{44}$$

The training procedure with SGD method can be summarized as follows

1. Select x randomly in $(0,1)$
2. Compute $\frac{error1(x)}{du_i} = 2 \left(\sum_j u_j B''_{j,p}(x) - f(x) \right) B''_{j,p}(x)$,
3. Compute $\frac{error2(x)}{du_i} = \left(\sum_j u_j B'_{j,p}(0) - g \right) B'_{j,p}(x)$,
4. Compute $\frac{error3(x)}{du_i} = \left(\sum_j u_j B_{j,p}(1) \right) B_{j,p}(x)$,
5. Compute $\frac{\partial error3(1)}{\partial u_i}$
6. Correct $u_i = u_i - \eta * \frac{\partial error(x)}{\partial u_i}$ where $error(x) = error1(x) + error2(x) + error3(x)$

for $\eta \in (0,1)$.

Another possibilites include defining

$$y = u(x) = \sum_i w_i \sigma \left(\sum_k u_k B_{k,p}(x) \right) \tag{45}$$

and repeating this recursively, or collecting several linear combinations into a vector

$$\begin{bmatrix} y_1 \\ \cdots \\ y_N \end{bmatrix} = \mathbf{u}(\mathbf{x}) = \begin{bmatrix} \sum_i w_i^1 \sigma \left(\sum_k u_k B_{k,p}(x) \right) \\ \sum_i w_i^2 \sigma \left(\sum_k u_k B_{k,p}(x) \right) \\ \cdots \\ \sum_i w_i^N \sigma \left(\sum_k u_k B_{k,p}(x) \right) \end{bmatrix} \tag{46}$$

or some other possibilities along these lines.

References

1. Cottrell, John Austin, Hughes, Thomas J. R., Bazilevs, Yuri, *Isogeometric Analysis: Towards Unification of Computer Aided Design and Finite Element Analysis* John Wiley and Sons, (2009).
2. Maziar, Raissi and Perdikaris, Paris and Karniadakis, George E., *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics, 378 (2019) 686–707
3. Bottou Léon, *Online Algorithms and Stochastic Approximations.* Online Learning and Neural Networks (1998) Cambridge University Press
4. Doległo, Kamil, Paszyńska, Anna, Paszyński, Maciej and Demkowicz, Leszek, Deep neural networks for smooth approximation of physics with higher order and continuity B-spline base functions, arXiv:2201.00904 (2022) 1-44
5. Paszyński, Maciej, Classical and isogeometric finite element method, AGH University, *https://epodreczniki.open.agh.edu.pl/handbook/31/module/1080/reader* [in Polish], *https://epodreczniki.open.agh.edu.pl/handbook/1088/module/1173/reader* [in English]