Pracownia problemowa 1

Maciej Paszyński[1], Anna Paszyńska[2]

[1] *Institute of Computer Science,*
AGH University of Science and Technology, Kraków, Poland
e-mail: maciej.paszynski@agh.edu.pl
[2] *Faculty of Physics, Astronomy and Applied Computer Science,*
Jagiellonian University, Kraków, Poland
e-mail: anna.paszynska@uj.edu.pl

Abstract

Considering the problem described in this document, for the modified three neural networks with k layers, with sigmoid activation function

$$ANN_1(n) = a_1\sigma\left(a_2\sigma\left(\cdots\sigma\left(a_k n + b_k\right)\cdots\right) + b_2\right) + b_1 \tag{1}$$

$$ANN_2(n) = c_1\sigma\left(c_2\sigma\left(\cdots\sigma\left(c_k n + d_k\right)\cdots\right) + d_2\right) + d_1 \tag{2}$$

$$ANN_3(n) = e_1\sigma\left(e_2\sigma\left(\cdots\sigma\left(e_k n + f_k\right)\cdots\right) + f_2\right) + f_1 \tag{3}$$

- Please define three loss functions (called also the error functions (13),(14))
- Plese compute the derivatives of the loss functions with respect to the neural network parameters (15-18)
- Please implement the three training procedures (20-23) and section 1.4
- Please plot the convergence of selected coefficients (Figure 4)
- Please plot the convergence of the error of the training procedure (Figure 5)
- Please compute the minimum and maximum difference between the trained neural network and dataset (like Section 1.5 for i=1,2,3)

Please prepare the raport with

- The formula for Artificial Neural Networks (like (11-12)
- The loss function (lie 13-14)

- The pseudo-code for the training procedure (like pseudo-codes in section 1.3)
- The MATLAB code (like section 1.4)
- The convergence of selected coefficients (like Figure 4)
- The convergence of error of loss function (like Figure 5)
- The value of minimum and maximum difference between the ANN and dataset (like Section 1.5 for i=1,2,3)

for $k = 1, 2, 3, ...$ (to see the convergence)

---

1. One-dimensional example of neural network learning coefficients of B-splines

1.1. One dimensional heat-transfer problem

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

$$B_{1,2}(x) = (1-x)^2; \quad B_{2,2}(x) = 2x(1-x); \quad B_{3,2}(x) = x^2 \tag{4}$$

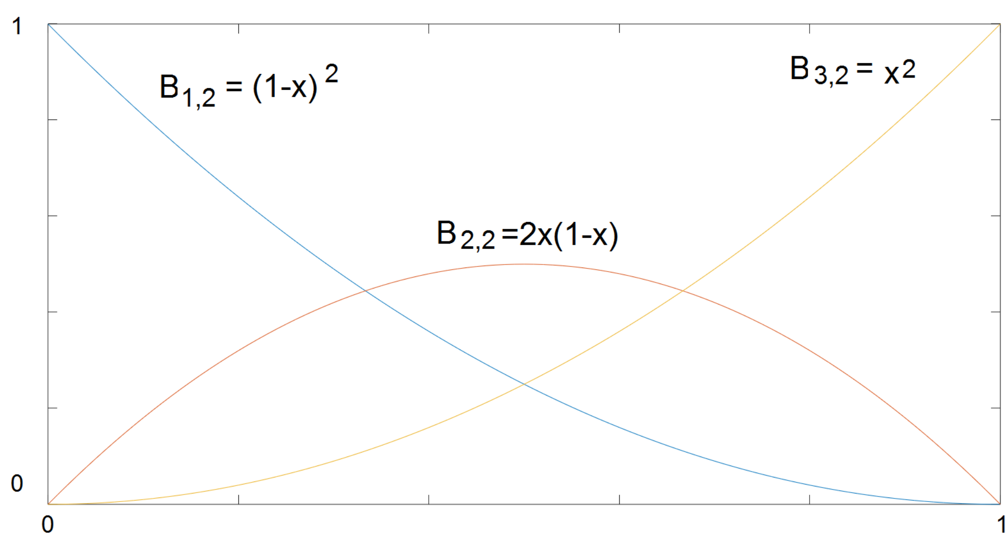

Figure 1: Three B-splines over a single interval (element)

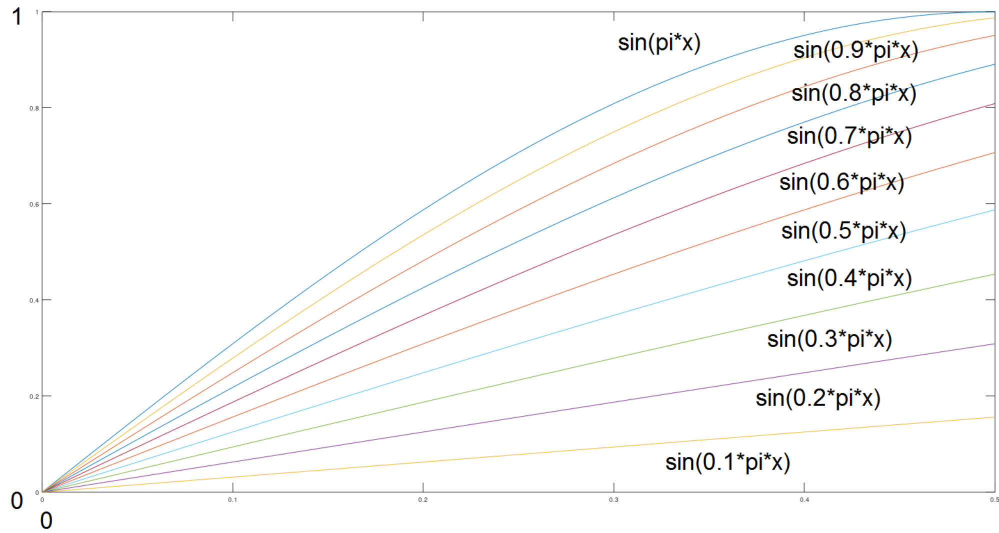

Figure 2: Plot of different solutions $u_n(x)$ for $n = 0.1, 0.2, ..., 1$.

Let us introduce the problem

$$-u''(x) = f(x) \quad x \in (0,1) \tag{5}$$

defined over $x \in (0,1)$, with boundary conditions $u(0) = 0$ and $u'(1) = g(x)$. 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

$$u_n(x) = sin(n\pi x) \tag{6}$$

We transform this problem into the weak form

$$\int_0^1 u'(x)v'(x)dx = \int_0^1 f(x)v(x)dx + v(1)g(1) \quad \forall v \tag{7}$$

and we discretize with B-spline basis functions

$$u_h = \sum_{i=1,2,3} u_i B_{i,2}(x) \tag{8}$$

to obtain

$$\begin{bmatrix} \int_{0,1} B'_{1,2}(x)B'_{1,2}(x)dx & \int_{0,1} B'_{1,2}(x)B'_{2,2}(x)dx & \int_{0,1} B'_{1,2}(x)B'_{3,2}(x)dx \\ \int_{0,1} B'_{2,2}(x)B'_{1,2}(x)dx & \int_{0,1} B'_{2,2}(x)B'_{2,2}(x)dx & \int_{0,1} B'_{2,2}(x)B'_{3,2}(x)dx \\ \int_{0,1} B'_{3,2}(x)B'_{1,2}(x)dx & \int_{0,1} B'_{3,2}(x)B'_{2,2}(x)dx & \int_{0,1} B'_{3,2}(x)B'_{3,2}(x)dx \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \int_{0,1} B_{1,2}(x)f_n(x)dx \\ \int_{0,1} B_{2,2}(x)f_n(x)dx \\ \int_{0,1} B_{3,2}(x)f_n(x)dx + n\pi cos(n\pi 1) \end{bmatrix} \tag{9}$$

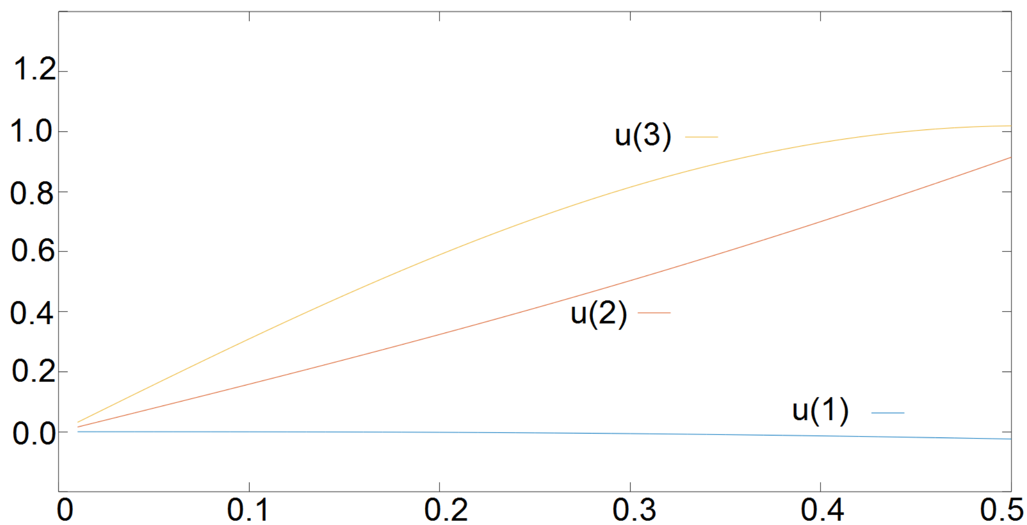

Figure 3: Coefficients of approximation $u_1 B_{1,2}(x) + u_2 B_{2,2} + u_3 B_{3,2}$ for $n \in (0, 0.5) \subset \mathcal{R}$.

1.2. Artificial neural network for u_i

Let us introduce the artificial neural network

$$ANN_i(n) = u_i \tag{10}$$

where n is the index of the f_n function, $i = 1, 2, 3$ (for three coefficients of B-splines).

Given sinus family function index n, it returns the coefficient u_i of B-splines for approximation of this function over $(0, 0.5)$.

$$ANN_i(n) = c_i \sigma\left(a_i n + b_i\right) + d_i \tag{11}$$

where the activation function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{12}$$

1.3. Training

The goal of the training is to find values of the weights a_i, b_i, c_i, d_i

We prepare a set of samples

- We randomly select $n \in (0, 1)$
- We solve the IGA problem (9)
- Input data (n), output data (u_1, u_2, u_3)

In other words, we train the neural network for some selected functions f_n, with a hope, that it will work for a given function of interest from the family.

How to train the artificial neural network? We define the error function

$$e_i(n) = 0.5\left(ANN_i(n) - u_i(n)\right)^2 = 0.5\left(c_i\sigma\left(a_i n + b_i\right) + d_i - u_i(n)\right)^2 = \tag{13}$$

$$0.5\left(\left(\frac{c_i}{1 + exp(-\left(a_i n + b_i\right)} + d_i\right) - u_i(n)\right)^2 \tag{14}$$

Now, we compute the derivatives

$$\frac{\partial e_i(n)}{\partial a_i} = \frac{c_i nexp(-a_i n - b_i)(ANN_i(n) - u_i(n))}{(exp(-ax - b) + 1)^2} \tag{15}$$

$$\frac{\partial e_i(n)}{\partial b_i} = \frac{c_i exp(-a_i n - b_i)(ANN_i(n) - u_i(n))}{(exp(-ax - b) + 1)^2} \tag{16}$$

$$\frac{\partial e_i(n)}{\partial c_i} = \frac{(ANN_i(n) - u_i(n))}{(exp(-ax - b) + 1)} \tag{17}$$

$$\frac{\partial e_i(n)}{\partial d_i} = (ANN_i(n) - u_i(n)) \tag{18}$$

$$\tag{19}$$

they say "how fast the error is changing if I modify a given coefficient".

We loop through the data set $\{n, (u_1(n), u_2(n), u_3(n))\}_{n\in A}$ where A is the set of selected points from $(0, 0.5)$, and we train each of the three ANN_1, ANN_2, and ANN_3

1. Select $(n, (u_1, u_2, u_3))$
2. Compute $u_i = ANN_i(n) = c_i\sigma\left(a_i n + b_i\right) + d_i$
3. Compute $e_i(n)$
4. Compute $\frac{\partial e_i(n)}{\partial a_i}, \frac{\partial e_i(n)}{\partial b_i}, \frac{\partial e_i(n)}{\partial c_i}, \frac{\partial e_i(n)}{\partial d_i}$

5. Correct

$$a_i = a_i - \eta * \frac{\partial e_i(n)}{\partial a_i} \tag{20}$$
$$b_i = b_i - \eta * \frac{\partial e_i(n)}{\partial b_i} \tag{21}$$
$$c_i = c_i - \eta * \frac{\partial e_i(n)}{\partial c_i} \tag{22}$$
$$d_i = d_i - \eta * \frac{\partial e_i(n)}{\partial d_i} \tag{23}$$

where $\eta \in (0, 1)$. This is like a local gradient method.

1.4. MATLAB implementation

```
   % Creation of dataset
% Here we solve the projection of the known solution u=sin(n*pi*x)
A = [1/5 1/10 1/30; 1/10 2/15 1/10; 1/30 1/10 1/5];
i=1;
for n=0.01:0.01:0.5
rhs= [ (pi*pi*n*n+2*cos(pi*n)-2)/(pi*pi*pi*n*n*n);
(-2*pi*n*sin(pi*n)-4*cos(pi*n)+4)/(pi*pi*pi*n*n*n);
((2-pi*pi*n*n)*cos(pi*n)+2*pi*n*sin(pi*n)-2)/(pi*pi*pi*n*n*n) ];
u=A \ rhs;
dataset_in(i)=n;
dataset_u1(i)=u(1);
dataset_u2(i)=u(2);
dataset_u3(i)=u(3);
i=i+1;
endfor
ndataset=i-1;
   % Training
a1=1.0; b1=1.0; c1=1.0; d1=1.0;
eta1=0.1;
r = 0 + (1-0).*rand(ndataset,1);
r=r.*ndataset;
for j=1:ndataset
i=floor(r(j));
eval1 = c1*1.0/(1.0+exp(-(a1*dataset_in(i)+b1)))+d1;
error1 = 0.5*(eval1-dataset_u1(i))^2;
```

```
derrorda = c1*dataset_in(i)*exp(-a1*dataset_in(i)-b1)*
(eval1-dataset_u1(i))/(exp(-a1*dataset_in(i)-b1)+1)^2;
a1=a1-eta1* derrorda;
derrordb = c1*exp(-a1*dataset_in(i)-b1)*
(eval1-dataset_u1(i))/(exp(-a1*dataset_in(i)-b1); b1=b1-eta1* derrordb;
derrordc = (eval1-dataset_u1(i))/(exp(-a1*dataset_in(i)-b1)+1);
c1=c1-eta1* derrordc;
derrordd = (eval1-dataset_u1(i));
d1=d1-eta1* derrordd;
```

We tried starting points $1.0, 10.0, -1.0, -10.0$ for all the combinations of a_i, b_i, c_i, d_i (256 runs) and the best result (smaller errors) we obtain for

$a_1 = b_1 = c_1 = d_1 = 1.0$; $a_2 = b_2 = 1, c_2 = 10.0, d_2 = -1.0$ $a_3 = b_2 = 3, c_3 = 10.0, d_3 = -1.0$

We used $\eta = 0.1$. We coded the ANN and the training in hand-made MATLAB code.

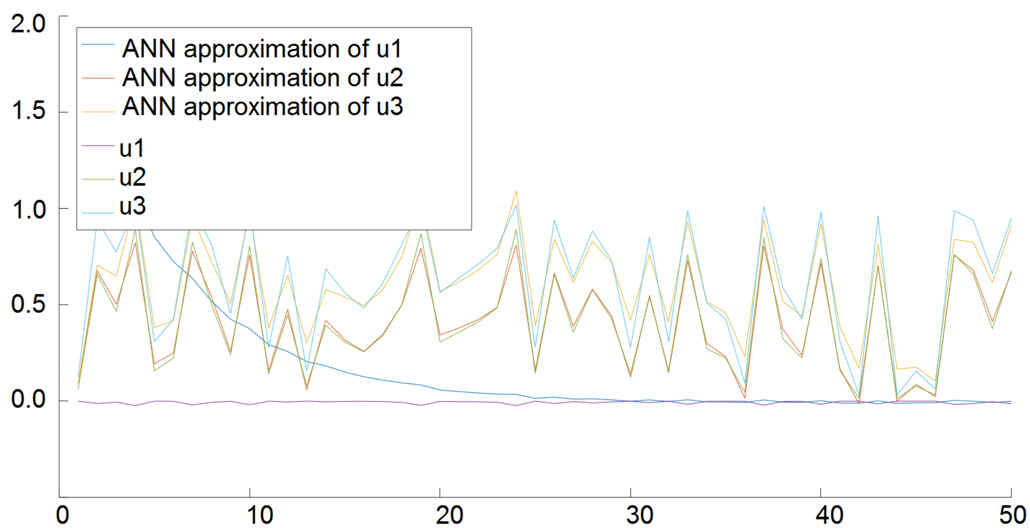

Figure 4: Training for the simple artificial nerual network (10) starting from $a_1 = b_1 = c_1 = d_1 = 1.0$; $a_2 = b_2 = 1, c_2 = 10.0, d_2 = -1.0$, $a_3 = b_2 = 3, c_3 = 10.0, d_3 = -1.0$, for $\eta = 0.1$.

1.5. Verification

We iterate through n from the range of $n \in (0, 1)$ and we compute the maximum and minimum difference between the ANN and correct values

Figure 5: Convergence of errors for training of ANN1, ANN2, ANN3

```
% Computing maximum and minimum difference for u(1)
i=1;
max_diff = 0; min_diff=1000;
for i=1:ndataset
n=dataset_in(i);
max_diff = max(max_diff,abs(dataset_u1(i)-c1*1.0/(1.0+exp(-(a1*n+b1)))+d1));

min_diff = min(min_diff,abs(dataset_u1(i)-c1*1.0/(1.0+exp(-(a1*n+b1)))+d1));

endfor
max_diff
min_diff
```

We obtain e.g.

```
max_diff = 0.2967
min_diff = 3.1143e-04
```