Pracownia Problemowa 1

Maciej Sikora
Patryk Krukowski

1 Ogólny wstęp

1.1 Pochodne funkcji straty

Odpowiednie pochodne funkcji straty zostały policzone symbolicznie w kodzie Matlaba, który zamieścimy w niniejszej sekcji w dalszym ciągu raportu.

2 Pierwsza zmodyfikowana sieć neuronowa

W metodzie tej będziemy przybliżać w sposób bezpośredni kombinację liniową B-spline'ów

$$y(x) = u_1 B_{1,2}(x) + u_2 B_{2,2}(x) + u_3 B_{3,2}(x)$$

ze współczynnikami rzeczywistymi u_1, u_2 oraz u_3, biorąc jako funkcję aproksymującą naszą zmodyfikowaną sieć, tzn.

$$ANN(x) = c_1\sigma(c_2\sigma(\dots c_k\sigma(a_{k1} + a_{k2}x + b_k) + d_k)\dots + d_2) + d_1.$$

Modyfikacja polega na tym, że uwzględniamy w sieci k warstw. Poniższe wyniki przedstawimy dla $k = 3$.

2.1 Funkcja straty

Funkcję straty definiujemy następująco:

$$error_1(x, a, b, c, d) = 0.5 * (ANN(x) - (u_1 * B_{1,2}))^2$$

$$error_2(x,a,b,c,d) = 0.5 * (ANN(x) - (u_2 * B_{2,2}))^2$$

$$error_3(x,a,b,c,d) = 0.5 * (ANN(x) - (u_3 * B_{3,2}))^2$$

2.2 Pseudokod procesu uczenia

1. Dane wejściowe - liczba warstw.
2. Generujemy dataset składający się z generacji $x \in (0, 0.5)$ oraz $n = 0.44$.
3. Korzystając ze znanego rozwiązania problemu IGA, generujemy współczynniki u_1, u_2, u_3.
4. Dla każdej warstwy sieci:
   (a) Liczymy $ANN(x)$.
   (b) Liczymy $e_1(x)$, $e_2(x)$, $e_3(x)$.
   (c) Liczymy pochodne cząstkowe pierwszego rzędu z $e(x)$ po każdym parametrze.
   (d) Uczymy parametry stochastycznym spadkiem wzdłuż gradientu, biorąc stałą uczącą $\eta \in (0, 1)$.

2.3 Kod Matlaba

```
function ANN_1(layers_number)
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Preparation of dataset

%collection of dataset
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) ];
A;
rhs;
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;
end
ndataset=i-1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%training

vec_a1=ones(1, layers_number);
vec_b1=ones(1, layers_number);
vec_c1=ones(1, layers_number);
vec_d1=zeros(1, layers_number);

vec_a2=ones(1, layers_number);
vec_b2=ones(1, layers_number);
vec_c2=ones(1, layers_number);
vec_d2=zeros(1, layers_number);

vec_a3=ones(1, layers_number);
vec_b3=ones(1, layers_number);
vec_c3=ones(1, layers_number);
vec_d3=zeros(1, layers_number);

```

```
41 smallest_err_u1=1000;
42 smallest_err_u2=1000;
43 smallest_err_u3=1000;
44
45
46 biggest_err_u1=0;
47 biggest_err_u2=0;
48 biggest_err_u3=0;
49
50 eta1=0.1;
51 eta2=0.1;
52 eta3=0.1;
53 r = 0 + (1-0).*rand(ndataset,1);
54 r=r.*ndataset;
55 n=0.444;
56 for idx=1:layers_number
57     %Symbolic functions
58
59     %Symbolic sigmoid
60     syms z a1 b1
61     a=sym('a', [1, idx]);
62     b=sym('b', [1, idx]);
63     c=sym('c', [1, idx]);
64     d=sym('d', [1, idx]);
65
66     sigmoid(z,a1,b1) = a1/(1+exp(-z))+b1;
67
68     result=sigmoid(z*a(idx)+b(idx), c(idx), d(idx));
69     if(idx>1)
70         for l=1:idx-1
71             result=sigmoid(a(idx-l)*result+b(idx-l), c
                (idx-l), d(idx-l));
72         end
```

```
73      end
74      ann_3(z,a,b,c,d)=result;
75
76
77      %Symbolic first and second derivative of PINN with
            respect to x
78
79      %Symbolic PDE solver function
80      temp=[a, b, c, d];
81      combined=temp(:);
82      combined=num2cell(combined);
83
84      %Symbolic error1 function
85      error1(z, a, b, c, d)=0.5 * (  ...
86        ann_3(z,combined{:})  ...
87        - ( ...
88          ((pi*pi*n*n+2*cos(pi*n)-2)/(pi*pi*pi*n*n*n)) *
                (1-z)^2 ...
89          ) ...
90        )^2;
91
92      error2(z, a, b, c, d)=0.5 * ( ...
93        ann_3(z,combined{:})  ...
94        - ( ...
95          ((-2*pi*n*sin(pi*n)-4*cos(pi*n)+4)/(pi*pi*pi*n
              *n*n)) * 2*z*(1-z) ...
96          ) ...
97        )^2;
98
99      error3(z, a, b, c, d)=0.5 * ( ...
100       ann_3(z,combined{:})  ...
101       - ( ...
102         (((2-pi*pi*n*n)*cos(pi*n)+2*pi*n*sin(pi*n)-2)
```

```
            /(pi*pi*pi*n*n*n)) * z^2 ...
103           ) ...
104         )^2;
105
106
107     for j=1:ndataset
108       params_u1(1:idx)=vec_a1(1:idx);
109       params_u1(idx+1:2*idx)=vec_b1(1:idx);
110       params_u1(2*idx+1:3*idx)=vec_c1(1:idx);
111       params_u1(3*idx+1:4*idx)=vec_d1(1:idx);
112
113       params_u2(1:idx)=vec_a2(1:idx);
114       params_u2(idx+1:2*idx)=vec_b2(1:idx);
115       params_u2(2*idx+1:3*idx)=vec_c2(1:idx);
116       params_u2(3*idx+1:4*idx)=vec_d2(1:idx);
117
118       params_u3(1:idx)=vec_a3(1:idx);
119       params_u3(idx+1:2*idx)=vec_b3(1:idx);
120       params_u3(2*idx+1:3*idx)=vec_c3(1:idx);
121       params_u3(3*idx+1:4*idx)=vec_d3(1:idx);
122
123       params_u1_combined=num2cell(params_u1);
124       params_u2_combined=num2cell(params_u2);
125       params_u3_combined=num2cell(params_u3);
126       i=floor(r(j));
127       if(i==0)
128         i=1;
129       end
130 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
131           %Approximation of u1, u2 and u3 coefficients
132       eval_u1 = double(ann_3(dataset_in(i),
              params_u1_combined{:}));
133       eval_u2 = double(ann_3(dataset_in(i),
```

```
        params_u2_combined{:}));
134     eval_u3 = double(ann_3(dataset_in(i),
        params_u3_combined{:}));
135
136     %Errors
137     error_u1 = 0.5*(eval_u1-dataset_u1(i))^2;
138     error_u2 = 0.5*(eval_u2-dataset_u2(i))^2;
139     error_u3 = 0.5*(eval_u3-dataset_u3(i))^2;
140     %Symbolic differentation
141     d_error_1_a(z, a, b, c, d)=diff(error1, a(idx));
142     d_error_1_b(z, a, b, c, d)=diff(error1, b(idx));
143     d_error_1_c(z, a, b, c, d)=diff(error1, c(idx));
144     d_error_1_d(z, a, b, c, d)=diff(error1, d(idx));
145
146     derror_u1_1a=double(d_error_1_a(dataset_in(i),
        params_u1_combined{:}));
147     derror_u1_1b=double(d_error_1_b(dataset_in(i),
        params_u1_combined{:}));
148     derror_u1_1c=double(d_error_1_c(dataset_in(i),
        params_u1_combined{:}));
149     derror_u1_1d=double(d_error_1_d(dataset_in(i),
        params_u1_combined{:}));
150
151     derror_u2_1a=double(d_error_1_a(dataset_in(i),
        params_u2_combined{:}));
152     derror_u2_1b=double(d_error_1_b(dataset_in(i),
        params_u2_combined{:}));
153     derror_u2_1c=double(d_error_1_c(dataset_in(i),
        params_u2_combined{:}));
154     derror_u2_1d=double(d_error_1_d(dataset_in(i),
        params_u2_combined{:}));
155
156     derror_u3_1a=double(d_error_1_a(dataset_in(i),
```

```
        params_u3_combined{:}));
157     derror_u3_1b=double(d_error_1_b(dataset_in(i),
        params_u3_combined{:}));
158     derror_u3_1c=double(d_error_1_c(dataset_in(i),
        params_u3_combined{:}));
159     derror_u3_1d=double(d_error_1_d(dataset_in(i),
        params_u3_combined{:}));
160
161     % Training of the boundary condition at x=0
162     d_error_2_a(z, a, b, c, d)=diff(error2, a(idx));
163     d_error_2_b(z, a, b, c, d)=diff(error2, b(idx));
164     d_error_2_c(z, a, b, c, d)=diff(error2, c(idx));
165     d_error_2_d(z, a, b, c, d)=diff(error2, d(idx));
166
167     derror_u1_2a=double(d_error_2_a(dataset_in(i),
        params_u1_combined{:}));
168     derror_u1_2b=double(d_error_2_b(dataset_in(i),
        params_u1_combined{:}));
169     derror_u1_2c=double(d_error_2_c(dataset_in(i),
        params_u1_combined{:}));
170     derror_u1_2d=double(d_error_2_d(dataset_in(i),
        params_u1_combined{:}));
171
172     derror_u2_2a=double(d_error_2_a(dataset_in(i),
        params_u2_combined{:}));
173     derror_u2_2b=double(d_error_2_b(dataset_in(i),
        params_u2_combined{:}));
174     derror_u2_2c=double(d_error_2_c(dataset_in(i),
        params_u2_combined{:}));
175     derror_u2_2d=double(d_error_2_d(dataset_in(i),
        params_u2_combined{:}));
176
177     derror_u3_2a=double(d_error_2_a(dataset_in(i),
```

```
        params_u3_combined{:}));
178     derror_u3_2b=double(d_error_2_b(dataset_in(i),
        params_u3_combined{:}));
179     derror_u3_2c=double(d_error_2_c(dataset_in(i),
        params_u3_combined{:}));
180     derror_u3_2d=double(d_error_2_d(dataset_in(i),
        params_u3_combined{:}));
181
182
183     % Training of the boundary condition at x=0.5
184     d_error_3_a(z, a ,b, c, d)=diff(error3, a(idx));
185     d_error_3_b(z, a, b, c, d)=diff(error3, b(idx));
186     d_error_3_c(z, a, b, c, d)=diff(error3, c(idx));
187     d_error_3_d(z, a, b, c, d)=diff(error3, d(idx));
188
189     derror_u1_3a=double(d_error_3_a(dataset_in(i),
        params_u1_combined{:}));
190     derror_u1_3b=double(d_error_3_b(dataset_in(i),
        params_u1_combined{:}));
191     derror_u1_3c=double(d_error_3_c(dataset_in(i),
        params_u1_combined{:}));
192     derror_u1_3d=double(d_error_3_d(dataset_in(i),
        params_u1_combined{:}));
193
194     derror_u2_3a=double(d_error_3_a(dataset_in(i),
        params_u2_combined{:}));
195     derror_u2_3b=double(d_error_3_b(dataset_in(i),
        params_u2_combined{:}));
196     derror_u2_3c=double(d_error_3_c(dataset_in(i),
        params_u2_combined{:}));
197     derror_u2_3d=double(d_error_3_d(dataset_in(i),
        params_u2_combined{:}));
198
```

```
199     derror_u3_3a=double(d_error_3_a(dataset_in(i),
            params_u3_combined{:}));
200     derror_u3_3b=double(d_error_3_b(dataset_in(i),
            params_u3_combined{:}));
201     derror_u3_3c=double(d_error_3_c(dataset_in(i),
            params_u3_combined{:}));
202     derror_u3_3d=double(d_error_3_d(dataset_in(i),
            params_u3_combined{:}));
203
204     vec_a1(idx)=vec_a1(idx) - eta1*(derror_u1_1a);
205     vec_b1(idx)=vec_b1(idx) - eta1*(derror_u1_1b);
206     vec_c1(idx)=vec_c1(idx) - eta1*(derror_u1_1c);
207     vec_d1(idx)=vec_d1(idx) - eta1*(derror_u1_1d);
208
209     vec_a2(idx)=vec_a2(idx) - eta2*(derror_u2_2a);
210     vec_b2(idx)=vec_b2(idx) - eta2*(derror_u2_2b);
211     vec_c2(idx)=vec_c2(idx) - eta2*(derror_u2_2c);
212     vec_d2(idx)=vec_d2(idx) - eta2*(derror_u2_2d);
213
214     vec_a3(idx)=vec_a3(idx) - eta3*(derror_u3_3a);
215     vec_b3(idx)=vec_b3(idx) - eta3*(derror_u3_3b);
216     vec_c3(idx)=vec_c3(idx) - eta3*(derror_u3_3c);
217     vec_d3(idx)=vec_d3(idx) - eta3*(derror_u3_3d);
218
219     if(error_u1 < smallest_err_u1)
220             smallest_err_u1=error_u1;
221             best_params_u1=num2cell(params_u1);
222             min_diff_u1=error_u1;
223         end
224
225         if(error_u1 > biggest_err_u1)
226             biggest_err_u1=error_u1;
227             max_diff_u1=error_u1;
```

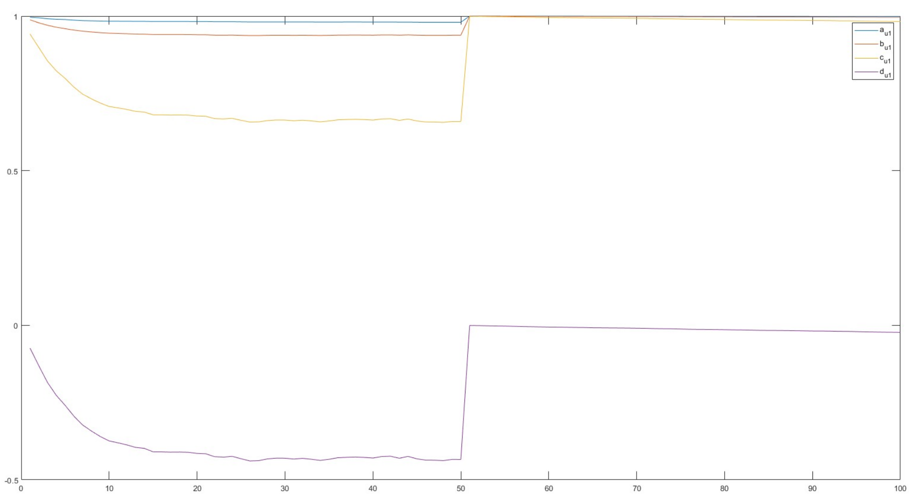
```
228         end
229
230         if(error_u2 < smallest_err_u2)
231             smallest_err_u2=error_u2;
232             best_params_u2=num2cell(params_u2);
233             min_diff_u2=error_u2;
234         end
235
236         if(error_u2 > biggest_err_u2)
237             biggest_err_u2=error_u2;
238             max_diff_u2=error_u2;
239         end
240
241         if(error_u3 < smallest_err_u3)
242             smallest_err_u3=error_u3;
243             best_params_u3=num2cell(params_u3);
244             min_diff_u3=error_u3;
245         end
246
247         if(error_u3 > biggest_err_u3)
248             biggest_err_u3=error_u3;
249             max_diff_u3=error_u3;
250         end
251
252     yeval_u1(j+(idx*ndataset-ndataset))=eval_u1;
253     yeval_u2(j+(idx*ndataset-ndataset))=eval_u2;
254     yeval_u3(j+(idx*ndataset-ndataset))=eval_u3;
255
256     ya_u1(j+(idx*ndataset-ndataset))=vec_a1(idx);
257     yb_u1(j+(idx*ndataset-ndataset))=vec_b1(idx);
258     yc_u1(j+(idx*ndataset-ndataset))=vec_c1(idx);
259     yd_u1(j+(idx*ndataset-ndataset))=vec_d1(idx);
260
```

```
    ya_u2(j+(idx*ndataset-ndataset))=vec_a2(idx);
    yb_u2(j+(idx*ndataset-ndataset))=vec_b2(idx);
    yc_u2(j+(idx*ndataset-ndataset))=vec_c2(idx);
    yd_u2(j+(idx*ndataset-ndataset))=vec_d2(idx);

    ya_u3(j+(idx*ndataset-ndataset))=vec_a3(idx);
    yb_u3(j+(idx*ndataset-ndataset))=vec_b3(idx);
    yc_u3(j+(idx*ndataset-ndataset))=vec_c3(idx);
    yd_u3(j+(idx*ndataset-ndataset))=vec_d3(idx);


    yy_u1(j+(idx*ndataset-ndataset))=dataset_u1(i);
    yy_u2(j+(idx*ndataset-ndataset))=dataset_u2(i);
    yy_u3(j+(idx*ndataset-ndataset))=dataset_u3(i);
    ye_u1(j+(idx*ndataset-ndataset))=error_u1
    ye_u2(j+(idx*ndataset-ndataset))=error_u2;
    ye_u3(j+(idx*ndataset-ndataset))=error_u3;
    j
  end
  idx
end
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot convergence
% plot convergence
x=1:layers_number*ndataset;
% hold on
% Vectors parameters  have parameters from every layer on the same plot!
plot(x,ya_u1,x,yb_u1,x,yc_u1,x,yd_u1);
h=legend('a_{u1}', 'b_{u1}', 'c_{u1}', 'd_{u1}');
set(h,'FontSize',10);
```

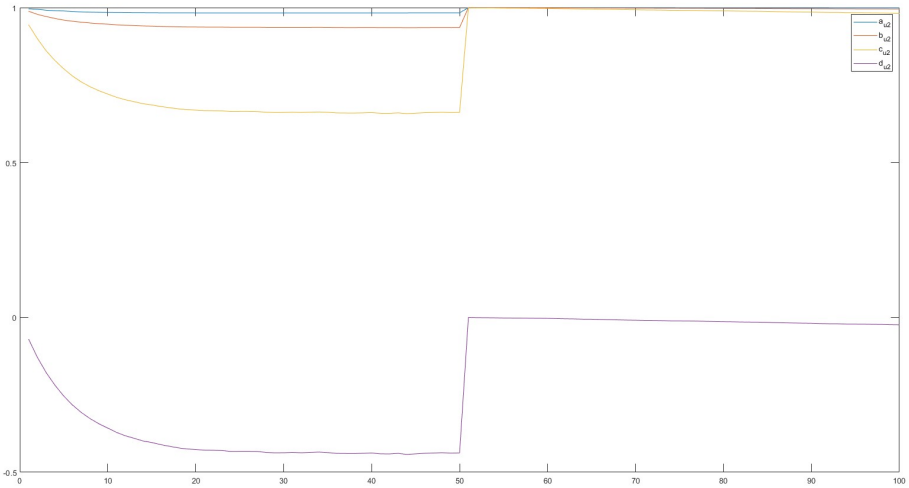
```
figure

plot(x,ya_u2,x,yb_u2,x,yc_u2,x,yd_u2);
h=legend('a_{u2}', 'b_{u2}', 'c_{u2}', 'd_{u2}');
set(h,'FontSize',10);
figure

plot(x,ya_u3,x,yb_u3,x,yc_u3,x,yd_u3);
h=legend('a_{u3}', 'b_{u3}', 'c_{u3}', 'd_{u3}');
set(h,'FontSize',10);

figure

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot error
x=1:1:layers_number*ndataset;
% hold on

plot(x,ye_u1,x,ye_u2,x,ye_u3);
h=legend('error(u1)', 'error(u2)', 'error(u3)');
set(h,'FontSize',20);
set(gca, 'YScale', 'log');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Computing maximum and minimum difference for u(1), u(2) and u(3)
max_diff_u1
min_diff_u1

```

```
max_diff_u2
min_diff_u2

max_diff_u3
min_diff_u3

%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

best_params_u1
best_params_u2
best_params_u3

end
```

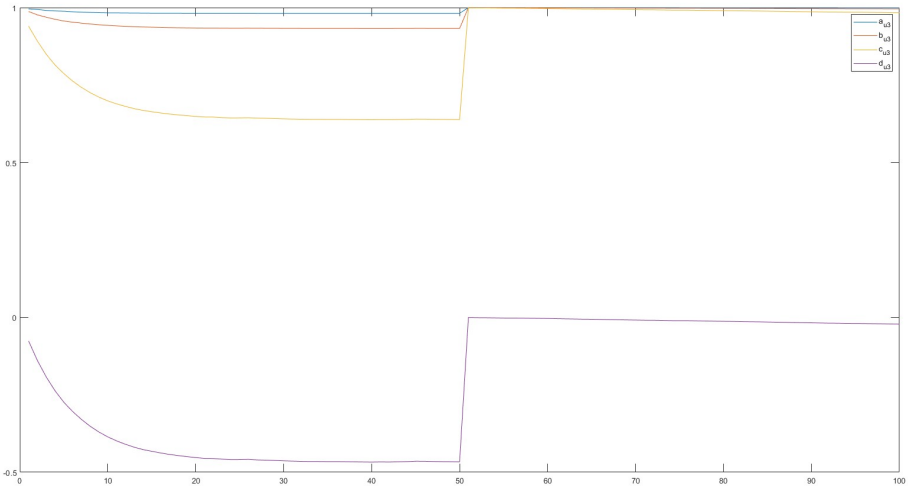
2.4 Zbieżność współczynników



Rysunek 1: Zbieżność uczonych parametrów u_1

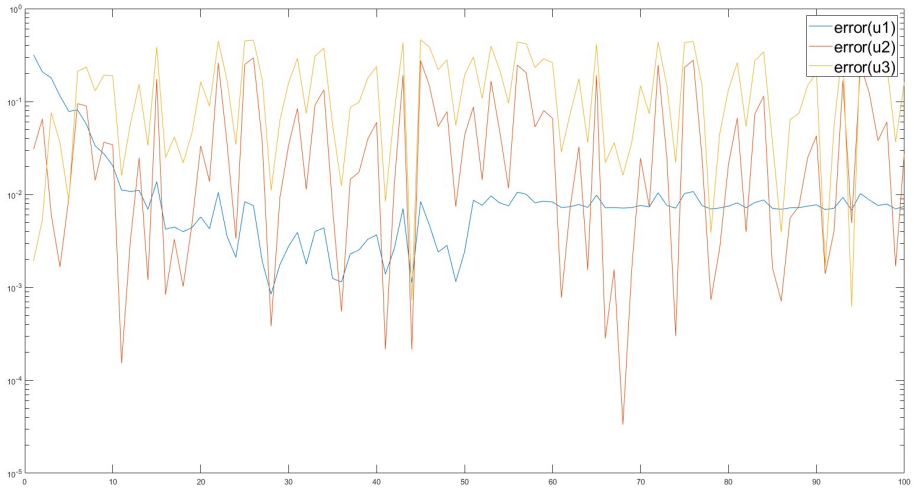


Rysunek 2: Zbieżność uczonych parametrów u_2



Rysunek 3: Zbieżność uczonych parametrów u_3

2.5 Zbieżność błędu funkcji straty



Rysunek 4: Funkcje błędu e_1, e_2, e_3

2.6 Maksymalna, minimalna wartość błędu oraz najlepsze parametry

$$max\_diff\_u1 = 0.3174$$

$$max\_diff\_u2 = 0.2956$$

$$max\_diff\_u3 = 0.4638$$

$$min\_diff\_u1 = 8.4484e-04$$

$$min\_diff\_u2 = 3.3425e-05$$

$$min\_diff\_u3 = 6.1907e-04$$

$$best\_params\_u1 => 0.9809, 0.9369, 0.6577, -0.4380$$

$$best\_params\_u2 => 0.9831, 0.9997, 0.9357, 0.9987, 0.6616, 0.9944, -0.4376, -0.0073$$

$$best\_params\_u3 => 0.9816, 0.9994, 0.9325, 0.9966, 0.6384, 0.9855, -0.4666, -0.0189$$

3 Druga zmodyfikowana sieć neuronowa

W metodzie tej będziemy przybliżać w sposób bezpośredni kombinację liniową B-spline'ów

$$y(n,x) = u_1(n)B_{1,2}(x) + u_2(n)B_{2,2}(x) + u_3(n)B_{3,2}(x)$$

ze współczynnikami rzeczywistymi u_1, u_2 oraz u_3, biorąc jako funkcję aproksymującą naszą zmodyfikowaną sieć, tzn.

$$ANN(n,x) = c_1\sigma(c_2\sigma(\dots c_k\sigma(a_{k1}n + a_{k2}x + b_k) + d_k)\dots + d_2) + d_1.$$

Modyfikacja polega na tym, że uwzględniamy w sieci k warstw. Poniższe wyniki przedstawimy dla $k = 3$.

3.1 Funkcja straty

Funkcję straty definiujemy następująco:

$$e(n,x) = 0.5 \cdot (ANN(n,x) - y(n,x))^2.$$

3.2 Pochodne funkcji straty

Odpowiednie pochodne funkcji straty zostały policzone symbolicznie w kodzie Matlaba, który zamieścimy w niniejszej sekcji w dalszym ciągu raportu.

3.3 Pseudokod procesu uczenia

1. Dane wejściowe - liczba warstw.
2. Generujemy dataset składający się z generacji $n, x \in (0, 0.5)$.
3. Korzystając ze znanego rozwiązania problemu IGA, generujemy współczynniki u_1, u_2, u_3, a następnie $y(n,x)$.
4. Dla każdej warstwy sieci:
   (a) Liczymy $ANN(n,x)$.
   (b) Liczymy $e(n,x)$.

(c) Liczymy pochodne cząstkowe pierwszego rzędu z $e(n, x)$ po każdym parametrze.

(d) Uczymy parametry stochastycznym spadkiem wzdłuż gradientu, biorąc stałą uczącą $\eta \in (0, 1)$.

Kod Matlaba

```
function ANN_2(layers_number)

%Preparation of dataset

%collection of dataset
A = [1/5 1/10 1/30; 1/10 2/15 1/10; 1/30 1/10 1/5];
i=1;
for n=0.1:0.1:0.5
    for x=0.1:0.1:0.5
    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;
    y=u(1)*(1-x)^2+u(2)*2*x.*(1-x)+u(3)*x^2;
    dataset_in_n(i)=n;
    dataset_in_x(i)=x;
    dataset_y(i)=y;
    i=i+1;
    end
end
ndataset=i-1;

r = 0 + (1-0).*rand(ndataset,1);
r=r.*ndataset;
```

```
% Training

vec_a1=ones(1, layers_number);
vec_a2=ones(1, layers_number);
vec_b=ones(1, layers_number);
vec_c=zeros(1, layers_number);
vec_d=zeros(1, layers_number);

eta=0.1;
smallest_error=1000;
biggest_error=0;
for idx=1:layers_number
    %Symbolic functions

    %Symbolic sigmoid
    syms z n e f g h p
    a1=sym('a1', [1, idx]);
    a2=sym('a2', [1, idx]);
    b=sym('b', [1, idx]);
    c=sym('c', [1, idx]);
    d=sym('d', [1, idx]);

    sigmoid(z,n,e,f,g,h,p) = h/(1+exp(-z*e-n*f-g))+p;

    result=sigmoid(z,n, a1(idx), a2(idx), b(idx), c(
        idx), d(idx));
    if(idx>1)
        for l=1:idx-1
            result=sigmoid(result,n,a1(idx-l),a2(idx-l
                ),b(idx-l),c(idx-l),d(idx-l));
        end
    end
    ann_2(z,n,a1,a2,b,c,d)=result;
```

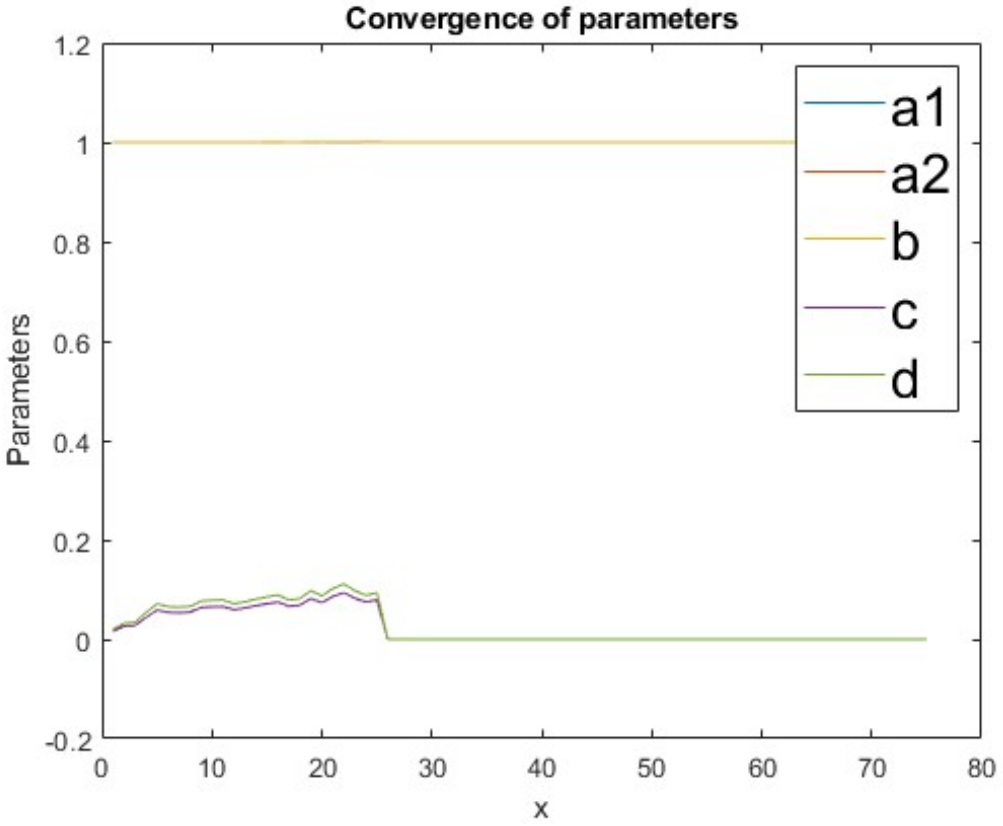
```
56
57      y=(pi*pi*n*n+2*cos(pi*n)-2)/(pi*pi*pi*n*n*n)*(1-z)
            ^2 + (-2*pi*n*sin(pi*n)-4*cos(pi*n)+4)/(pi*pi*
            pi*n*n*n)*(1-z) + ((2-pi*pi*n*n)*cos(pi*n)+2*pi
            *n*sin(pi*n)-2)/(pi*pi*pi*n*n*n)*z^2;
58      y(z,n)=y;
59      %Symbolic MSE
60      temp=[a1, a2, b, c, d];
61      combined=temp(:);
62      combined=num2cell(combined);
63      err(z,n,a1,a2,b,c,d)=0.5*(ann_2(z,n,combined{:}) -
            y(z,n))^2;
64
65      for j=1:ndataset
66        i=floor(r(j));
67
68        if(i==0)
69          i=1;
70        end
71
72       params(1:idx)=vec_a1(1:idx);
73       params(idx+1:2*idx)=vec_a2(1:idx);
74       params(2*idx+1:3*idx)=vec_b(1:idx);
75       params(3*idx+1:4*idx)=vec_c(1:idx);
76       params(4*idx+1:5*idx)=vec_d(1:idx);
77
78       params_combined=num2cell(params);
79
80       %Approximation of y
81       eval = double(ann_2(dataset_in_x(i), dataset_in_n
             (i), params_combined{:}));
82       error = 0.5*(eval-dataset_y(i))^2;
83
```

```
84      if(error < smallest_error)
85          smallest_error=error;
86          best_params=num2cell(params);
87      end
88
89      if(error > biggest_error)
90          biggest_error=error;
91      end
92
93      %Symbolic differentation
94      d_error_a1(z, n, a1, a2, b, c, d)=diff(err, a1(
            idx));
95      d_error_a2(z,n,a1,a2, b, c, d)=diff(err, a2(idx))
            ;
96      d_error_b(z,n,a1,a2, b, c, d)=diff(err, b(idx));
97      d_error_c(z,n,a1,a2, b, c, d)=diff(err, c(idx));
98      d_error_d(z,n,a1,a2, b, c, d)=diff(err, d(idx));
99
100     derror_a1=double(d_error_a1(dataset_in_x(i),
            dataset_in_n(i), params_combined{:}));
101     derror_a2=double(d_error_a2(dataset_in_x(i),
            dataset_in_n(i), params_combined{:}));
102     derror_b=double(d_error_b(dataset_in_x(i),
            dataset_in_n(i), params_combined{:}));
103     derror_c=double(d_error_c(dataset_in_x(i),
            dataset_in_n(i), params_combined{:}));
104     derror_d=double(d_error_d(dataset_in_x(i),
            dataset_in_n(i), params_combined{:}));
105
106     vec_a1(idx)=vec_a1(idx) - eta*derror_a1;
107     vec_a2(idx)=vec_a2(idx) - eta*derror_a2;
108     vec_b(idx)=vec_b(idx) - eta*derror_b;
109     vec_c(idx)=vec_c(idx) - eta*derror_c;
```

```
     vec_d(idx)=vec_d(idx) - eta*derror_d;

     yeval(j+(idx*ndataset-ndataset))=eval;
     yy(j+(idx*ndataset-ndataset))=dataset_y(i);
     ya1(j+(idx*ndataset-ndataset))=vec_a1(idx);
     ya2(j+(idx*ndataset-ndataset))=vec_a2(idx);
     yb(j+(idx*ndataset-ndataset))=vec_b(idx);
     yc(j+(idx*ndataset-ndataset))=vec_c(idx);
     yd(j+(idx*ndataset-ndataset))=vec_d(idx);
     ye(j+(idx*ndataset-ndataset))=error
     j

    end
    idx
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot convergence
x=1:ndataset*layers_number;

plot(x,ya1,x,ya2,x,yb,x,yc,x,yd);
h=legend('a1','a2','b','c','d');
set(h,'FontSize',20);


figure

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot error
```

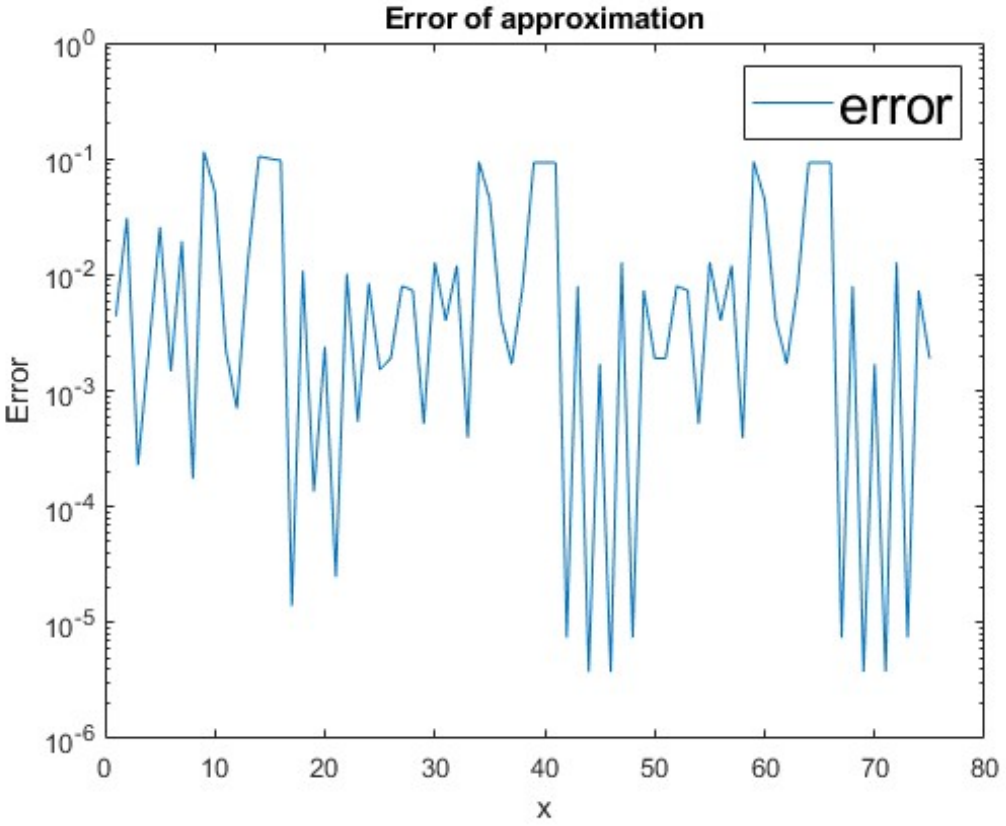
```matlab
x=1:layers_number*ndataset;
% hold on

plot(x,ye);
h=legend('error');
set(h,'FontSize',20);
set(gca, 'YScale', 'log');

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The biggest and the smallest error, and the best parameters from all
% layers

biggest_error
smallest_error
best_params

end
```

3.4 Zbieżność współczynników



Rysunek 5: Zbieżność uczonych parametrów

3.5 Zbieżność błędu funkcji straty



Rysunek 6: Zbieżność błędu funkcji straty

3.6 Maksymalna, minimalna wartość błędu oraz najlepsze parametry

- Największy błąd: 0.1153
- Najmniejszy błąd: $3.7660e^{-6}$

4 Trzecia zmodyfikowana sieć neuronowa

W metodzie tej będziemy przybliżać współczynniki kombinacji liniowej u_1, u_2 oraz u_3 poprzez użycie Physics Informed Neural Network. Poniższe wyniki przedstawimy dla trzech warstw.

4.1 Funkcja straty

Funkcję straty dla i-tego współczynnika u_i definiujemy następująco:

$$e_{1,i}(x) = 0.5 \cdot (PINN_{xx,i}(x) + n^2\pi^2 \sin(n\pi x))^2.$$

Definiujemy jeszcze funkcje straty zgodne z warunkami Dirichleta i von Neumanna: $e_{2,i}(\cdot) = 0.5e_{1,i}(0), e_{3,i}(\cdot) = 0.5e_{1,i}(0.5)$.

4.2 Pochodne funkcji straty

Odpowiednie pochodne funkcji straty zostały policzone symbolicznie w kodzie Matlaba, który zamieścimy w niniejszej sekcji w dalszym ciągu raportu.

4.3 Pseudokod procesu uczenia

1. Dane wejściowe - liczba warstw.
2. Generujemy dataset składający się z generacji $x \in (0, 0.5)$.
3. Korzystając ze znanego rozwiązania problemu IGA, generujemy współczynniki u_1, u_2, u_3.
4. Dla każdej warstwy sieci oraz dla każdego współczynnika u_i:
   (a) Liczymy $PINN(x)$.
   (b) Liczymy $e_1(x)$, $e_2(\cdot)$ oraz $e_3(\cdot)$
   (c) Liczymy pochodne cząstkowe pierwszego rzędu z wyżej zdefiniowanych błędów po każdym parametrze.
   (d) Uczymy parametry stochastycznym spadkiem wzdłuż gradientu, biorąc stałą uczącą $\eta \in (0, 1)$.

Kod Matlaba

```
function ANN_3(layers_number)
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% 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;
end
ndataset=i-1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Training
%Initialization for every (three in this case) coefficients of a linear
%combination of B-spline functions
vec_a1=ones(1, layers_number);
vec_b1=ones(1, layers_number);
vec_c1=ones(1, layers_number);
vec_d1=zeros(1, layers_number);

vec_a2=ones(1, layers_number);
vec_b2=ones(1, layers_number);
vec_c2=ones(1, layers_number);
```

```
vec_d2=zeros(1, layers_number);

vec_a3=ones(1, layers_number);
vec_b3=ones(1, layers_number);
vec_c3=ones(1, layers_number);
vec_d3=zeros(1, layers_number);

smallest_err_u1=1000;
smallest_err_u2=1000;
smallest_err_u3=1000;

biggest_err_u1=0;
biggest_err_u2=0;
biggest_err_u3=0;

eta1=0.1;
eta2=0.1;
eta3=0.1;
r = 0 + (1-0).*rand(ndataset,1);
r=r.*ndataset;
n=0.444;
for idx=1:layers_number
    %Symbolic functions

    %Symbolic sigmoid
    syms z a1 b1
    a=sym('a', [1, idx]);
    b=sym('b', [1, idx]);
    c=sym('c', [1, idx]);
    d=sym('d', [1, idx]);

    sigmoid(z,a1,b1) = a1/(1+exp(-z))+b1;

```

```
64      result=sigmoid(z*a(idx)+b(idx), c(idx), d(idx));
65      if(idx>1)
66          for l=1:idx-1
67              result=sigmoid(a(idx-l)*result+b(idx-l), c
                    (idx-l), d(idx-l));
68          end
69      end
70      ann_3(z,a,b,c,d)=result;
71
72
73      %Symbolic first and second derivative of PINN with
            respect to x
74      ann_3_x(z,a,b,c,d)=diff(ann_3, z);
75      ann_3_xx(z,a,b,c,d)=diff(ann_3_x, z);
76
77      %Symbolic PDE solver function
78      temp=[a, b, c, d];
79      combined=temp(:);
80      combined=num2cell(combined);
81      F(z,a, b, c, d)=ann_3_xx(z, combined{:}) + n^2*pi
            ^2*sin(n*pi*z);
82
83      %Symbolic error1 function
84      error1(z, a, b, c, d)=0.5*F(z, combined{:})^2;
85
86      %Symbolic error2 function
87      error2(a, b, c, d)=0.5*ann_3(0, combined{:});
88
89      %Symbolic error3 function
90      error3(a, b, c, d)=0.5*(ann_3_x(0.5, combined{:})
            - n*pi*cos(n*pi*0.5))^2;
91
92      for j=1:ndataset
```

```
93          params_u1(1:idx)=vec_a1(1:idx);
94          params_u1(idx+1:2*idx)=vec_b1(1:idx);
95          params_u1(2*idx+1:3*idx)=vec_c1(1:idx);
96          params_u1(3*idx+1:4*idx)=vec_d1(1:idx);
97
98          params_u2(1:idx)=vec_a2(1:idx);
99          params_u2(idx+1:2*idx)=vec_b2(1:idx);
100         params_u2(2*idx+1:3*idx)=vec_c2(1:idx);
101         params_u2(3*idx+1:4*idx)=vec_d2(1:idx);
102
103         params_u3(1:idx)=vec_a3(1:idx);
104         params_u3(idx+1:2*idx)=vec_b3(1:idx);
105         params_u3(2*idx+1:3*idx)=vec_c3(1:idx);
106         params_u3(3*idx+1:4*idx)=vec_d3(1:idx);
107
108         params_u1_combined=num2cell(params_u1);
109         params_u2_combined=num2cell(params_u2);
110         params_u3_combined=num2cell(params_u3);
111
112         i=floor(r(j));
113         if(i==0)
114             i=1;
115         end
116
117         %Approximation of u1, u2 and u3 coefficients
118         eval_u1 = double(ann_3(dataset_in(i),
                params_u1_combined{:}));
119         eval_u2 = double(ann_3(dataset_in(i),
                params_u2_combined{:}));
120         eval_u3 = double(ann_3(dataset_in(i),
                params_u3_combined{:}));
121
122         %Errors
```

```
123         error_u1 = 0.5*(eval_u1-dataset_u1(i))^2;
124         error_u2 = 0.5*(eval_u2-dataset_u2(i))^2;
125         error_u3 = 0.5*(eval_u3-dataset_u3(i))^2;
126
127         %Symbolic differentation
128         d_error_1_a(z, a, b, c, d)=diff(error1, a(idx)
                );
129         d_error_1_b(z,a, b, c, d)=diff(error1, b(idx))
                ;
130         d_error_1_c(z, a, b, c, d)=diff(error1, c(idx)
                );
131         d_error_1_d(z, a, b, c, d)=diff(error1, d(idx)
                );
132
133         derror_u1_1a=double(d_error_1_a(dataset_in(i),
                params_u1_combined{:}));
134         derror_u1_1b=double(d_error_1_b(dataset_in(i),
                params_u1_combined{:}));
135         derror_u1_1c=double(d_error_1_c(dataset_in(i),
                params_u1_combined{:}));
136         derror_u1_1d=double(d_error_1_d(dataset_in(i),
                params_u1_combined{:}));
137
138         derror_u2_1a=double(d_error_1_a(dataset_in(i),
                params_u2_combined{:}));
139         derror_u2_1b=double(d_error_1_b(dataset_in(i),
                params_u2_combined{:}));
140         derror_u2_1c=double(d_error_1_c(dataset_in(i),
                params_u2_combined{:}));
141         derror_u2_1d=double(d_error_1_d(dataset_in(i),
                params_u2_combined{:}));
142
143         derror_u3_1a=double(d_error_1_a(dataset_in(i),
```

```
        params_u3_combined{:}));
144     derror_u3_1b=double(d_error_1_b(dataset_in(i),
        params_u3_combined{:}));
145     derror_u3_1c=double(d_error_1_c(dataset_in(i),
        params_u3_combined{:}));
146     derror_u3_1d=double(d_error_1_d(dataset_in(i),
        params_u3_combined{:}));
147
148     % Training of the boundary condition at x=0
149     d_error_2_a(a, b, c, d)=diff(error2, a(idx));
150     d_error_2_b(a, b, c, d)=diff(error2, b(idx));
151     d_error_2_c(a, b, c, d)=diff(error2, c(idx));
152     d_error_2_d(a, b, c, d)=diff(error2, d(idx));
153
154     derror_u1_2a=double(d_error_2_a(
           params_u1_combined{:}));
155     derror_u1_2b=double(d_error_2_b(
           params_u1_combined{:}));
156     derror_u1_2c=double(d_error_2_c(
           params_u1_combined{:}));
157     derror_u1_2d=double(d_error_2_d(
           params_u1_combined{:}));
158
159     derror_u2_2a=double(d_error_2_a(
           params_u2_combined{:}));
160     derror_u2_2b=double(d_error_2_b(
           params_u2_combined{:}));
161     derror_u2_2c=double(d_error_2_c(
           params_u2_combined{:}));
162     derror_u2_2d=double(d_error_2_d(
           params_u2_combined{:}));
163
164     derror_u3_2a=double(d_error_2_a(
```

```
    params_u3_combined{:}));
derror_u3_2b=double(d_error_2_b(
    params_u3_combined{:}));
derror_u3_2c=double(d_error_2_c(
    params_u3_combined{:}));
derror_u3_2d=double(d_error_2_d(
    params_u3_combined{:}));


% Training of the boundary condition at x=0.5
d_error_3_a(a,b, c, d)=diff(error3, a(idx));
d_error_3_b(a, b, c, d)=diff(error3, b(idx));
d_error_3_c(a,b, c, d)=diff(error3, c(idx));
d_error_3_d(a, b, c, d)=diff(error3, d(idx));

derror_u1_3a=double(d_error_3_a(
    params_u1_combined{:}));
derror_u1_3b=double(d_error_3_b(
    params_u1_combined{:}));
derror_u1_3c=double(d_error_3_c(
    params_u1_combined{:}));
derror_u1_3d=double(d_error_3_d((
    params_u1_combined{:})));

derror_u2_3a=double(d_error_3_a(
    params_u2_combined{:}));
derror_u2_3b=double(d_error_3_b(
    params_u2_combined{:}));
derror_u2_3c=double(d_error_3_c(
    params_u2_combined{:}));
derror_u2_3d=double(d_error_3_d(
    params_u2_combined{:}));

```

```
186         derror_u3_3a=double(d_error_3_a(
               params_u3_combined{:}));
187         derror_u3_3b=double(d_error_3_b(
               params_u3_combined{:}));
188         derror_u3_3c=double(d_error_3_c(
               params_u3_combined{:}));
189         derror_u3_3d=double(d_error_3_d(
               params_u3_combined{:}));
190
191         vec_a1(idx)=vec_a1(idx) - eta1*(derror_u1_3a+
               derror_u1_2a+derror_u1_1a);
192         vec_b1(idx)=vec_b1(idx) - eta1*(derror_u1_3b+
               derror_u1_2b+derror_u1_1b);
193         vec_c1(idx)=vec_c1(idx) - eta1*(derror_u1_3c+
               derror_u1_2c+derror_u1_1c);
194         vec_d1(idx)=vec_d1(idx) - eta1*(derror_u1_3d+
               derror_u1_2d+derror_u1_1d);
195
196         vec_a2(idx)=vec_a2(idx) - eta2*(derror_u2_3a+
               derror_u2_2a+derror_u2_1a);
197         vec_b2(idx)=vec_b2(idx) - eta2*(derror_u2_3b+
               derror_u2_2b+derror_u2_1b);
198         vec_c2(idx)=vec_c2(idx) - eta2*(derror_u2_3c+
               derror_u2_2c+derror_u2_1c);
199         vec_d2(idx)=vec_d2(idx) - eta2*(derror_u2_3d+
               derror_u2_2d+derror_u2_1d);
200
201         vec_a3(idx)=vec_a3(idx) - eta3*(derror_u3_3a+
               derror_u3_2a+derror_u3_1a);
202         vec_b3(idx)=vec_b3(idx) - eta3*(derror_u3_3b+
               derror_u3_2b+derror_u3_1b);
203         vec_c3(idx)=vec_c3(idx) - eta3*(derror_u3_3c+
               derror_u3_2c+derror_u3_1c);
```

```
204         vec_d3(idx)=vec_d3(idx) - eta3*(derror_u3_3d+
                derror_u3_2d+derror_u3_1d);
205
206         if(error_u1 < smallest_err_u1)
207             smallest_err_u1=error_u1;
208             best_params_u1=num2cell(params_u1);
209             min_diff_u1=error_u1;
210         end
211
212         if(error_u1 > biggest_err_u1)
213             biggest_err_u1=error_u1;
214             max_diff_u1=error_u1;
215         end
216
217         if(error_u2 < smallest_err_u2)
218             smallest_err_u2=error_u2;
219             best_params_u2=num2cell(params_u2);
220             min_diff_u2=error_u2;
221         end
222
223         if(error_u2 > biggest_err_u2)
224             biggest_err_u2=error_u2;
225             max_diff_u2=error_u2;
226         end
227
228         if(error_u3 < smallest_err_u3)
229             smallest_err_u3=error_u3;
230             best_params_u3=num2cell(params_u3);
231             min_diff_u3=error_u3;
232         end
233
234         if(error_u3 > biggest_err_u3)
235             biggest_err_u3=error_u3;
```

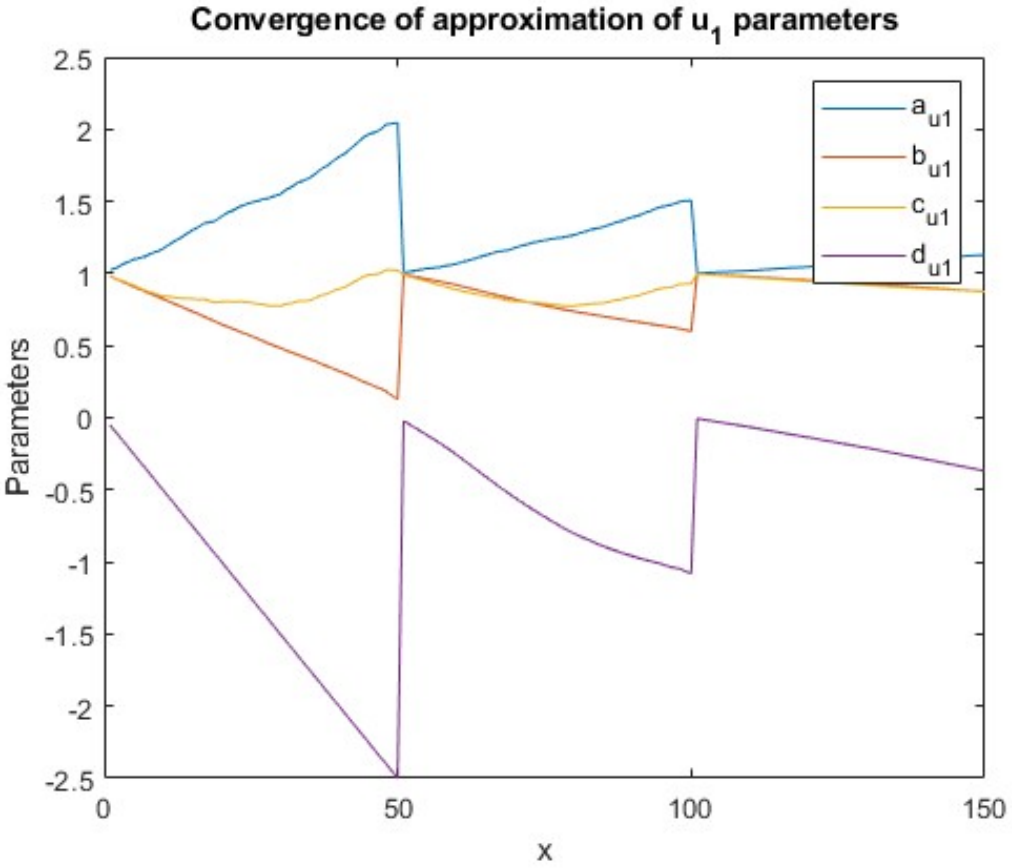
```
236             max_diff_u3=error_u3;
237         end
238
239         yeval_u1(j+(idx*ndataset-ndataset))=eval_u1;
240         yeval_u2(j+(idx*ndataset-ndataset))=eval_u2;
241         yeval_u3(j+(idx*ndataset-ndataset))=eval_u3;
242
243         ya_u1(j+(idx*ndataset-ndataset))=vec_a1(idx);
244         yb_u1(j+(idx*ndataset-ndataset))=vec_b1(idx);
245         yc_u1(j+(idx*ndataset-ndataset))=vec_c1(idx);
246         yd_u1(j+(idx*ndataset-ndataset))=vec_d1(idx);
247
248         ya_u2(j+(idx*ndataset-ndataset))=vec_a2(idx);
249         yb_u2(j+(idx*ndataset-ndataset))=vec_b2(idx);
250         yc_u2(j+(idx*ndataset-ndataset))=vec_c2(idx);
251         yd_u2(j+(idx*ndataset-ndataset))=vec_d2(idx);
252
253         ya_u3(j+(idx*ndataset-ndataset))=vec_a3(idx);
254         yb_u3(j+(idx*ndataset-ndataset))=vec_b3(idx);
255         yc_u3(j+(idx*ndataset-ndataset))=vec_c3(idx);
256         yd_u3(j+(idx*ndataset-ndataset))=vec_d3(idx);
257
258
259         yy_u1(j+(idx*ndataset-ndataset))=dataset_u1(i)
                ;
260         yy_u2(j+(idx*ndataset-ndataset))=dataset_u2(i)
                ;
261         yy_u3(j+(idx*ndataset-ndataset))=dataset_u3(i)
                ;
262         ye_u1(j+(idx*ndataset-ndataset))=error_u1
263         ye_u2(j+(idx*ndataset-ndataset))=error_u2;
264         ye_u3(j+(idx*ndataset-ndataset))=error_u3;
265         j
```

```
    end
    idx
end

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot convergence
x=1:layers_number*ndataset;
% hold on
% Vectors parameters  have parameters from every layer on the same plot!
plot(x,ya_u1,x,yb_u1,x,yc_u1,x,yd_u1);
h=legend('a_{u1}', 'b_{u1}', 'c_{u1}', 'd_{u1}');
set(h,'FontSize',10);
figure

plot(x,ya_u2,x,yb_u2,x,yc_u2,x,yd_u2);
h=legend('a_{u2}', 'b_{u2}', 'c_{u2}', 'd_{u2}');
set(h,'FontSize',10);
figure

plot(x,ya_u3,x,yb_u3,x,yc_u3,x,yd_u3);
h=legend('a_{u3}', 'b_{u3}', 'c_{u3}', 'd_{u3}');
set(h,'FontSize',10);

figure

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot error
x=1:1:layers_number*ndataset;
```

```matlab
% hold on

plot(x,ye_u1,x,ye_u2,x,ye_u3);
h=legend('error(u1)', 'error(u2)', 'error(u3)');
set(h,'FontSize',20);
set(gca, 'YScale', 'log');

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Computing maximum and minimum difference for u(1), u(2) and u(3)
max_diff_u1
min_diff_u1

max_diff_u2
min_diff_u2

max_diff_u3
min_diff_u3

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
best_params_u1
best_params_u2
best_params_u3
end
```

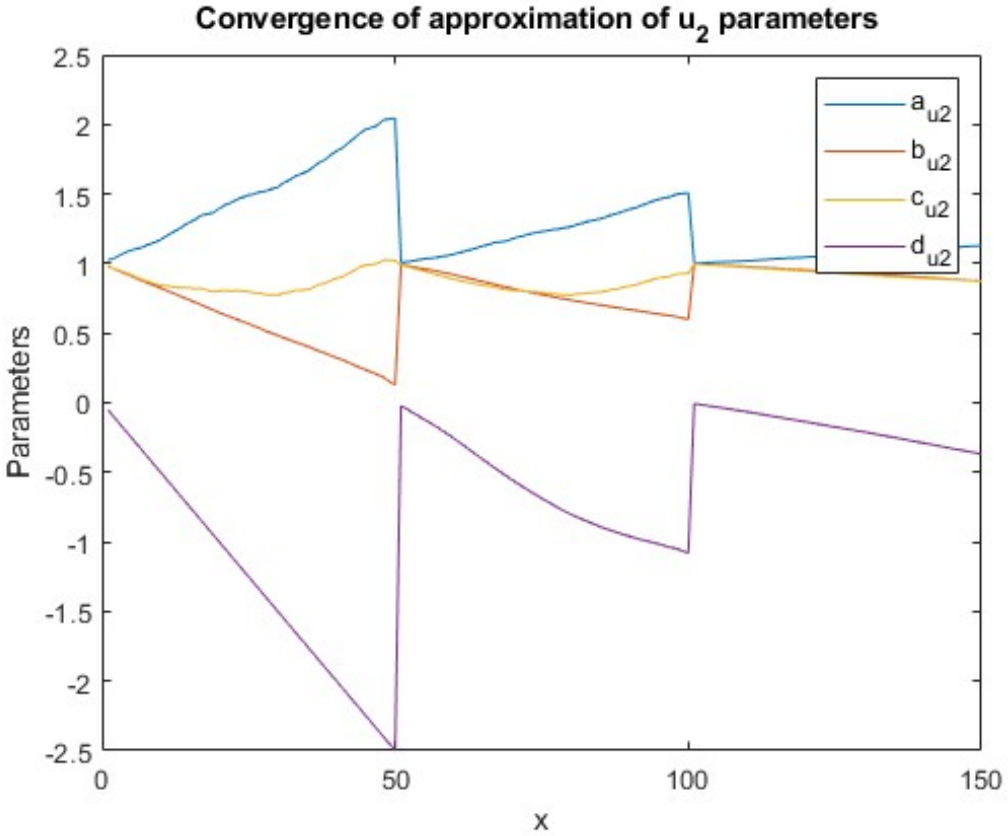
4.4 Zbieżność współczynników

- Dla u_1



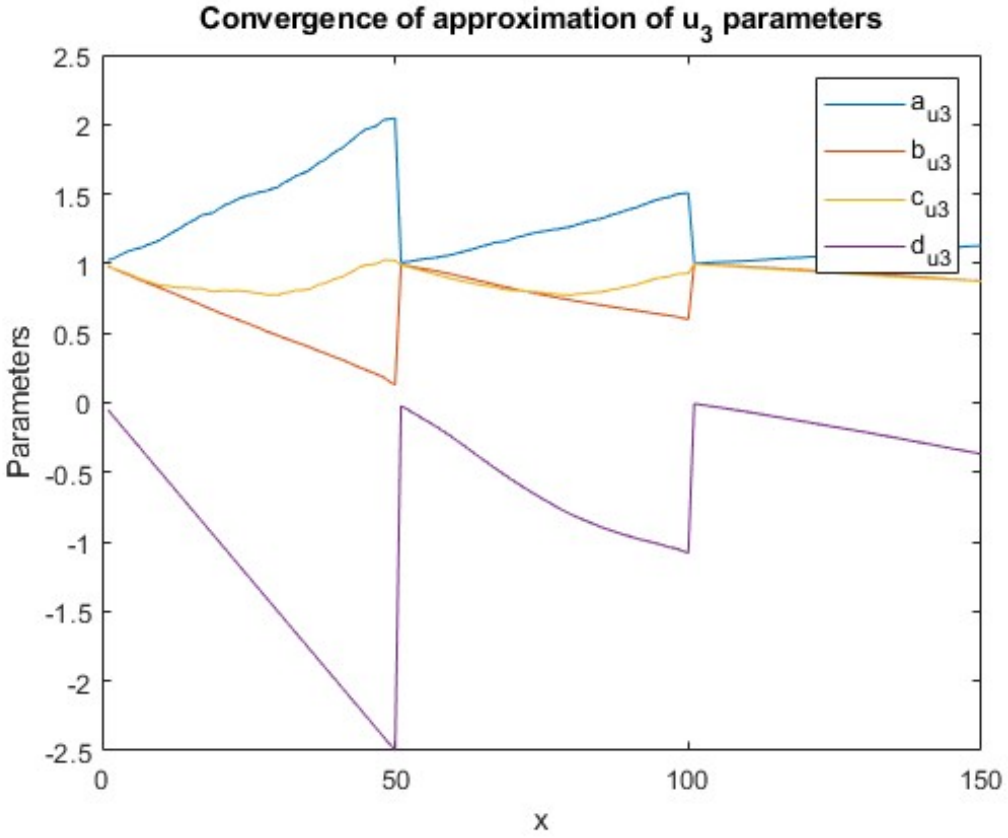
Rysunek 7: Zbieżność uczonych parametrów u_1

- Dla u_2



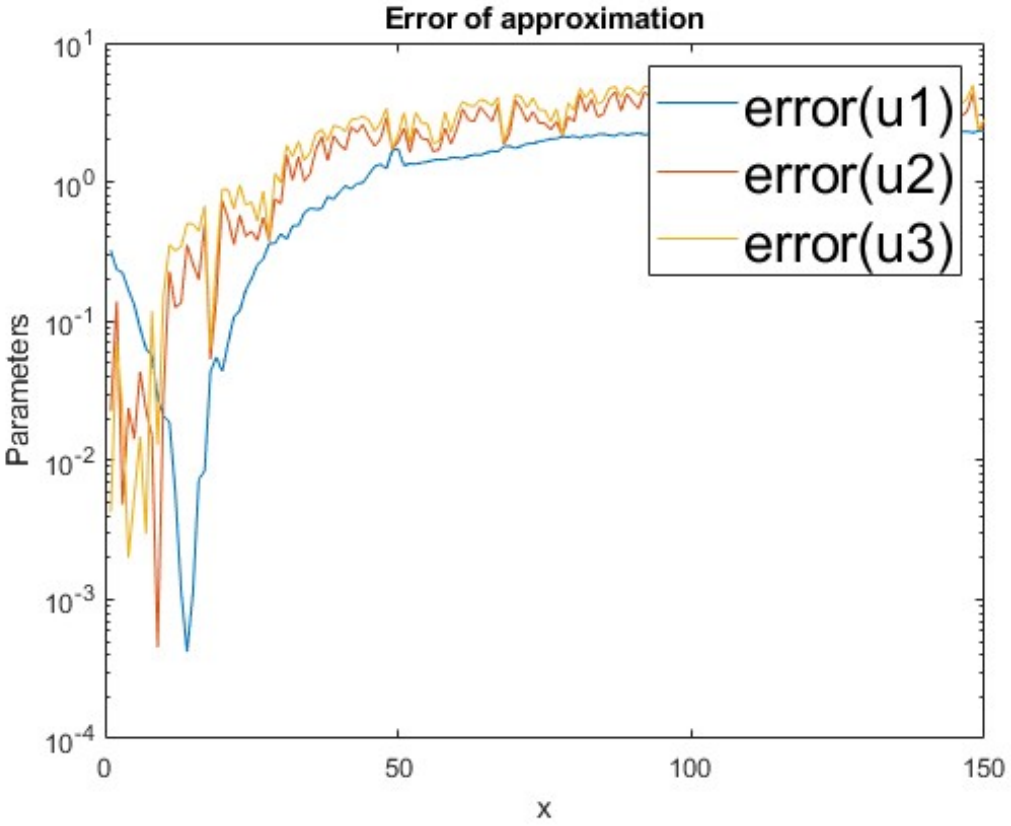
Rysunek 8: Zbieżność uczonych parametrów u_2

- Dla u_3



Rysunek 9: Zbieżność uczonych parametrów u_3

4.5 Zbieżność błędu funkcji straty



Rysunek 10: Zbieżność błędu

4.6 Maksymalna, minimalna wartość błędu

- Dla u_1
  - Największy błąd: 2.3717
  - Najmniejszy błąd: $4.2174e^{-4}$
- Dla u_2
  - Największy błąd: 4.5774
  - Najmniejszy błąd: $4.5347e^{-4}$
- Dla u_3

– Największy błąd: 4.9766

– Najmniejszy błąd: 0.0020