[CAD CAE] Lab 1

Autor rozwiązań: Maciej Sikora

Link do wersji webowej tego pdf: https://perfect-bill-43f.notion.site/CAD-CAE-Lab-1-fb7ba75d86b94068a1e238f91a1dde9b

Zmodyfikowany lekko kod udostępniony od prowadących

▼ Zmodyfikowany kod

```
% This subroutine draws B-Spline basis functions using given knot vector.
% How to run
%
% splines(number of points to draw, knot vector as a matrix)
%
% Examples
%
% draws evenly distributed knots, 1-st order splines, 3 elements
% splines (1000, [1,1,2,3,4,4])
%
% draws unevenly distributed knots, 3-rd order splines, 6 elements
% splines (100000,[0,0,0,0,1,3,6,10,11,11.5,11.5,11.5,11.5])
%
% draws evenly distributed knots with repeated knots, 2-nd order splines, 3 elements
% splines (1000, [1,1,1,2,2,3,4,4,4])
function splines_comp(precision,knot_vector,coeffs)

% subroutine calculating number of basis functions
compute_nr_basis_functions = @(knot_vector,p) size(knot_vector, 2) - p - 1;
% subroutine generating mesh for drawing basis functions
mesh = @(a,c) [a:(c-a)/precision:c];
% subroutine drawing basis functions
plot_spline = @(knot_vector,p,nr,x) plot(x,compute_splines(knot_vector,p,nr,x));

% computing order of polynomials
p = compute_p(knot_vector);
% validation of knot vector construction
t = check_sanity(knot_vector,p);
% if knot vector is poorly constructed - stop further processing
if (~t)
  disp("Poorly constructed knot_vector")
  return
end
% computating number of basis functions
nr = compute_nr_basis_functions(knot_vector,p);

% beginning of drawing range
x_begin = knot_vector(1);
% end of drawing range
x_end = knot_vector(size(knot_vector,2));
x=mesh(x_begin,x_end);

z=zeros(1,precision+1);
add_splines(knot_vector,p,nr,x,z,coeffs(1));
for i=1:nr
  z=add_splines(knot_vector,p,i,x,z,coeffs(i));
```

```
end


img = imread('obraz.jpg');
image('CData',img,'XData',[1 40],'YData',[1 0])
hold on
plot(x,z,'r');
axis([1 40 0 1])
hold off



% Subroutine computing order of polynomials
function p=compute_p(knot_vector)

% first entry in knot_vector
  initial = knot_vector(1);
% lenght of knot_vector
  kvsize = size(knot_vector,2);
  p = 0;

% checking number of repetitions of first entry in knot_vector
  while (p+2 <= kvsize) && (initial == knot_vector(p+2))
    p = p+1;
  end

  return
end

% Subroutine computing basis functions according to recursive Cox-de-Boor formulae
function y=compute_splines(knot_vector,p,nr,x)

y=compute_spline(knot_vector,p,nr,x);
return
end

% Subroutine computing basis functions according to recursive Cox-de-Boor formulae
function y=add_splines(knot_vector,p,nr,x,z,coeff)

ybuff=coeff*compute_spline(knot_vector,p,nr,x);
y=ybuff+z;
return
end



% Subroutine computing basis functions according to recursive Cox-de-Boor formulae
function y=compute_spline(knot_vector,p,nr,x)

% function (x-x_i)/(x_{i-p}-x_i)
  fC= @(x,a,b) (x-a)/(b-a);
% function (x_{i+p+1}-x)/(x_{i+p+1}-x_{i+1})
  fD= @(x,c,d) (d-x)/(d-c);

% x_i
  a = knot_vector(nr);
% x_{i-p}
  b = knot_vector(nr+p);
% x_{i+1}
  c = knot_vector(nr+1);
% x_{i+p+1}
  d = knot_vector(nr+p+1);

% linear function for p=0
```

```
  if (p==0)
    y = 0 .* (x < a) + 1 .* (a <= x & x <= d) + 0 .* (x > d);
    return
  end

% B_{i,p-1}
  lp = compute_spline(knot_vector,p-1,nr,x);
% B_{i+1,p-1}
  rp = compute_spline(knot_vector,p-1,nr+1,x);

% (x-x_i)/(x_{i-p)-x_i)*B_{i,p-1}
  if (a==b)
% if knots in knot_vector are repeated we have to include it in formula
    y1 = 0 .* (x < a) + 1 .* (a <= x & x <= b) + 0 .* (x > b);
  else
    y1 = 0 .* (x < a) + fC(x,a,b) .* (a <= x & x <= b) + 0 .* (x > b);
  end

% (x_{i+p+1}-x)/(x_{i+p+1)-x_{i+1})*B_{i+1,p-1}
  if (c==d)
% if knots in knot_vector are repeated we have to include it in formula
    y2 = 0 .* (x < c) + 1 .* (c < x & x <= d) + 0 .* (d < x);
  else
    y2 = 0 .* (x < c) + fD(x,c,d) .* (c < x & x <= d) + 0 .* (d < x);
  end

  y = lp .* y1 + rp .* y2;
  return
end


% Subroutine checking sanity of knot_vector
function t=check_sanity(knot_vector,p)

  initial = knot_vector(1);
  kvsize = size(knot_vector,2);

  t = true;
  counter = 1;

% if number of repeated knots at the beginning of knot_vector doesn't match polynomial order
  for i=1:p+1
    if (initial ~= knot_vector(i))
% return FALSE
      t = false;
      return
    end
  end

% if there are too many repeated knots in the middle of knot_vector
  for i=p+2:kvsize-p-1
    if (initial == knot_vector(i))
      counter = counter + 1;
      if (counter > p)
% return FALSE
        t = false;
        return
      end
    else
      initial = knot_vector(i);
      counter = 1;
    end
  end
```

```
  initial = knot_vector(kvsize);

% if number of repeated knots at the end of knot_vector doesn't match polynomial order
  for i=kvsize-p:kvsize
    if (initial ~= knot_vector(i))
% return FALSE
      t = false;
      return
    end
  end

% if subsequent element in knot_vector is smaller than previous one
  for i=1:kvsize-1
    if (knot_vector(i)>knot_vector(i+1))
% return FALSE
      t = false;
      return
    end
  end

  return

end



end
```

▼ Kody pomocnicze

Kod do generowania wstępnych węzłów

```
from random import randint

def get_diff_rand():
    return (randint(0, 100)-50) / 1000

knots = 80
tier = 4
weights = [0.9 + get_diff_rand() for _ in range(knots - tier + 1)]

knot = []

for i in range(1, int((knots-1)/(tier-2)+1)):
    for _ in range(tier-2):
        knot.append(i)

knot.insert(0,1)
knot.append(knots//(tier-2)-1)

print(f"splines_comp(10000, {knot}, {weights})")
```

Kod do kopiowania wyniku

```
import os
def addToClipBoard(text):
    command = 'echo ' + text.strip() + '| clip'
    os.system(command)

with open('odp.txt', 'r') as file:
    text = file.read().replace('\n', '')
    print(text)
    addToClipBoard(text)
```

▼ Odpowiedź

knots =

[1, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10, 11, 11, 12, 12, 13, 13, 14, 14, 15, 15, 16, 16, 17, 17, 18, 18, 19, 19, 20, 20, 21, 21, 22, 22, 23, 23, 24, 24, 25, 25, 26, 26, 27, 27, 28, 28, 29, 29, 30, 30, 31, 31, 32, 32, 33, 33, 34, 34, 35, 35, 36, 36, 37, 37, 38, 38, 39, 39, 39]

weights =

[0.83, 0.83, 0.83, 0.82, 0.831, 0.842, 0.86, 0.864, 0.88, 0.86, 0.89, 0.892, 0.89, 0.89, 0.89, 0.885, 0.88, 0.86, 0.847, 0.845, 0.849, 0.85, 0.858, 0.872, 0.878, 0.877, 0.876, 0.874, 0.875, 0.87, 0.87, 0.87, 0.8589, 0.88, 0.9, 0.92, 0.93, 0.94, 0.95, 0.95, 0.94, 0.938, 0.925, 0.92, 0.91,
0.9, 0.88, 0.87, 0.865, 0.86, 0.85, 0.84, 0.858, 0.865, 0.86, 0.85, 0.84, 0.83, 0.82, 0.81, 0.8, 0.79, 0.78, 0.77, 0.76, 0.75, 0.74, 0.73, 0.757, 0.763, 0.79, 0.79, 0.75, 0.74, 0.73, 0.72, 0.71]

▼ Screen wynikowego obrazka