

Melléklet

A. Varioshore TPU, flow-hőmérséklet mérési adatok

Első iteráció													
Hőm	Beállított flow	Nominális	1	2	3	4	5	6	7	8	Átlag	Arány	Beállítandó flow
214	1.0000	0.5	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51	0.51	1.0200	0.9804
215	0.9000	0.5	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.9400	0.9574
216	0.8000	0.5	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.9200	0.8696
223	0.5000	0.5	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.48	0.9600	0.5208

Második iteráció													
Hőm	Beállított flow	Nominális	1	2	3	4	5	6	7	8	Átlag	Arány	Beállítandó flow
213	0.9900	0.5	0.5	0.5	0.5	0.5	0.49	0.49	0.49	0.49	0.495	0.9900	1.0000
217	0.7100	0.5	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.7800	0.9103
221	0.5700	0.5	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.8600	0.6628
232	0.4500	0.5	0.45	0.45	0.45	0.45	0.45	0.46	0.46	0.46	0.4538	0.9075	0.4959

Harmadik iteráció													
Hőm	Beállított flow	Nominális	1	2	3	4	5	6	7	8	Átlag	Arány	Beállítandó flow
215	1.0000	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1.0000	1.0000
220	0.7200	0.5	0.49	0.49	0.49	0.48	0.48	0.48	0.48	0.48	0.4838	0.9675	0.7442
230	0.5000	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1.0000	0.5000
240	0.4700	0.5	0.5	0.5	0.5	0.49	0.49	0.51	0.51	0.51	0.5013	1.0025	0.4688

B. Görbe és felület illesztés a mért hőmérséklet-flow rate adatokra

```

data1 = ( {
    {212, 1.01},
    {220, 0.6415},
    {228, 0.4981},
    {236, 0.4317}
} );
data2 = ( {
    {214, 0.9677},
    {219, 0.617},
    {225, 0.4694},
    {231, 0.4494}
} );

model3 = Ph + (Ph - Pl) Tanh[\[Gamma]1 (T - Tg)];
fit3 = NonlinearModelFit[
    data1, {model3, 0 < Ph < 1, 0 < \[Gamma]1 < 1}, {Ph,
    Pl, \[Gamma]1, {Tg, 220}}, T]
fit4 = NonlinearModelFit[
    data2, {model3, 0 < Ph < 1, 0 < \[Gamma]1 < 1}, {Ph,
    Pl, \[Gamma]1, {Tg, 220}}, T]
fit3["RSquared"]
fit3["BestFitParameters"]
fit4["BestFitParameters"]
Show[Plot[{fit3[t], fit4[t]}, {t, 210, 240},
    Epilog -> Map[Point, data1] ], ListPlot[data2],
    PlotRange -> {0.4, 1}]

Ph\[Rho][\[Rho]_] := mPh \[Rho] + bPh
Pl\[Rho][\[Rho]_] := mPl \[Rho] + bPl
\[Gamma]\[Rho][\[Rho]_] := m\[Gamma] \[Rho] + b\[Gamma]
Tg\[Rho][\[Rho]_] := mTg \[Rho] + bTg
PhSolve =
    Solve[Ph\[Rho][5] == Evaluate[Ph /. fit3["BestFitParameters"]
    ]] &&
    Ph\[Rho][3] == Evaluate[Ph /. fit4["BestFitParameters"]]]
PlSolve =
    Solve[Pl\[Rho][5] == Evaluate[Pl /. fit3["BestFitParameters"]
    ]] &&

```

```

P1\[Rho][3] == Evaluate[P1 /. fit4["BestFitParameters"]]]
\[Gamma]Solve =
Solve\[Gamma]\[Rho][5] ==
    Evaluate\[Gamma]1 /.
        fit3["BestFitParameters"]]] && \[Gamma]\[Rho][3] ==
        Evaluate\[Gamma]1 /. fit4["BestFitParameters"]]]
TgSolve =
Solve[Tg\[Rho][5] == Evaluate[Tg /. fit3["BestFitParameters"]
]] &&
    Tg\[Rho][3] == Evaluate[Tg /. fit4["BestFitParameters"]]]

modelVegleges[T_, f_] :=
    Ph\[Rho][
        f] + (Ph\[Rho][f] -
            P1\[Rho][f]) Tanh\[Gamma]\[Rho][f](T - Tg\[Rho][f]))

mPh = mPh /. First@PhSolve;
bPh = bPh /. First@PhSolve;
bP1 = bP1 /. First@P1Solve;
mP1 = mP1 /. First@P1Solve;
b\[Gamma] = b\[Gamma] /. First@\[Gamma]Solve;
m\[Gamma] = m\[Gamma] /. First@\[Gamma]Solve;
bTg = bTg /. First@TgSolve;
mTg = mTg /. First@TgSolve;

Show[Plot3D[modelVegleges[T, \[Rho]], {T, 210, 240}, {\[Rho],
    2, 6},
    Mesh -> {8, 3}, MeshStyle -> {Black, {Red, Thickness ->
        0.006}},
    ColorFunction -> (ColorData[{"LightTemperatureMap",
        "Reverse"}][#3] &),
    LabelStyle -> {FSZ, FontFamily -> FFAM, Black},
    AxesLabel -> {"Nyomtatasihomerseklet, T[°C]",
        "Terfogataram, q[!(*SuperscriptBox[(mm),
            \[3)])!(*SuperscriptBox[(s), 1(-1)])!]", "Flowrate
            [1]"},
    ImageSize -> 120 mm],
    ListPointPlot3D[dataFinal,
    PlotStyle -> {Directive[PointSize[0.025], Black]}]]

```

C. Posztprocesszor, alkalmazás

```
1 import sys
2 import customtkinter
3 import constants as c
4 import func as f
5 from PIL import Image
6 from PIL import ImageTk
7 import os
8
9 # G-code forrásfájl elérési útvonala, ami a parancssori argumentumként
   lesz megadva
10 c.sourceFile = sys.argv[1]
11
12 # A program belépési pontja
13 f.start_routine()
14
15 class App(customtkinter.CTk):
16     def __init__(self):
17         super().__init__()
18
19         # Az alkalmazás ablakának beállításai
20         self.geometry("750x510")
21         self.title("Hab posztprocesszor")
22         self.grid_columnconfigure((0,1), weight=1)
23         self.grid_rowconfigure(0, weight=1)
24         self.grid_rowconfigure(1, weight=7)
25         self.grid_propagate(False)
26         self.grid_columnconfigure(0, weight=0)
27         self.grid_rowconfigure(0, weight=0)
28
29         # Frame-ek létrehozása
30         self.frame_1 = customtkinter.CTkFrame(master=self, width=450,
height=450)
31         self.frame_1.grid(row=0, column=0, rowspan=2, pady=10, padx
=(10,0), sticky="nwes")
32         self.frame_1.grid_rowconfigure(0, weight=0)
33         self.frame_1.grid_columnconfigure(0, weight=0)
34
35         self.frame_2 = customtkinter.CTkFrame(master=self, width=250,
height=250)
36         self.frame_2.grid(row=0, column=1, rowspan=1, pady=(10,0), padx
=10, sticky="nwes")
37         self.frame_2.grid_rowconfigure(0, weight=1)
38         self.frame_2.grid_columnconfigure(0, weight=1)
39         self.frame_2.grid_propagate(False)
40
41         self.frame_3 = customtkinter.CTkFrame(master=self, width=250,
```

```
height=120)
42     self.frame_3.grid(row=1, column=1, rowspan=1, pady=10, padx=10,
sticky="nwes")
43     self.frame_3.grid_rowconfigure(0, weight=1)
44     self.frame_3.grid_rowconfigure(1, weight=1)
45     self.frame_3.grid_columnconfigure(0, weight=1)
46     self.frame_3.grid_propagate(False)
47
48     # Képek hozzáadása a frame-ekhez
49     self.label1 = customtkinter.CTkLabel(master=self.frame_1, image=
f.front_view(c.color), text="")
50     self.label1.grid(row=0, column=0, padx=10, pady=10)
51
52     iso_im = customtkinter.CTkImage(Image.open(os.path.join(c.
current_path, 'partim.jpg')), size=(230,230))
53     self.label2 = customtkinter.CTkLabel(master=self.frame_2, image=
iso_im, text="")
54     self.label2.grid(row=0, column=0, padx=10, pady=10)
55
56     # "Mentés" gomb
57     self.button = customtkinter.CTkButton(master=self.frame_3, text=
"Mentés", command=self.mentes)
58     self.button.grid(row=0, column=0, pady=(10,0), padx=10, sticky='
we', columnspan=3)
59
60     # Menü
61     optionmenu_var = customtkinter.StringVar(value="Homogén
habosítás")
62     self.optionmenu = customtkinter.CTkOptionMenu(self.frame_3,
values=["Homogén habosítás", "Gradiens habosítás", "Homogén, lépcss
habosítás"], command=self.optionmenu_callback, variable=
optionmenu_var)
63     self.optionmenu.grid(row=1, column=0, padx=10, pady=10,
columnspan=3, sticky="we")
64
65     # Slider1 beállítása
66     self.slider1 = customtkinter.CTkSlider(self.frame_3, from_=45,
to=100, number_of_steps=55, command=self.slider1_event, width=180)
67     self.slider1.grid(row=2, column=0, padx=10, pady=10, sticky="ew"
)
68     self.slider1.set(100)
69
70     self.label_slider1 = customtkinter.CTkLabel(self.frame_3, text="
Relatív srség: 100%")
71     self.label_slider1.grid(row=2, column=1, padx=(0,10))
72
73     # Anyagválasztó
74     anyag_var = customtkinter.StringVar(value="LW-PLA")
```

```
75     self.optionmenu_anyag = customtkinter.CTkOptionMenu(self.frame_1
, values=["LW-PLA", "Variosshore TPU"], command=self.anyag_menu,
variable=anyag_var)
76     self.optionmenu_anyag.grid(row=1, column=0, padx=10, pady=(0,10)
)
77
78     # Lista és gombok a Z magasságok és relatív srségek
beállításához
79     self.z_height_entries = []
80     self.rel_den_sliders = []
81     self.z_height_labels = []
82
83     def add_entry(self):
84         # Új bejegyzés hozzáadása a Z magasság és relatív srség
beállításához
85
86         self.label_text = str(1 + len(self.z_height_entries)) + ". Z
módosítás:"
87
88         z_height_label = customtkinter.CTkLabel(self.frame_3, text=self.
label_text)
89         z_height_label.grid(row=3 + len(self.z_height_entries), column
=0, padx=10, pady=10)
90
91         z_height_entry = customtkinter.CTkEntry(self.frame_3)
92         z_height_entry.grid(row=3 + len(self.z_height_entries), column
=1, padx=0, pady=10)
93
94         rel_den_slider = customtkinter.CTkSlider(self.frame_3, from_=45,
to=100, number_of_steps=55, width=180)
95         rel_den_slider.grid(row=3 + len(self.rel_den_sliders), column=2,
padx=10, pady=10)
96         rel_den_slider.set(100)
97
98         rel_den_slider.configure(command=lambda value=rel_den_slider,
label=self.rel_den_label: self.update_rel_den_label(value, label))
99
100        self.z_height_entries.append(z_height_entry)
101        self.rel_den_sliders.append(rel_den_slider)
102        self.z_height_labels.append(z_height_label)
103
104        self.window_height = 455 + 55 * len(self.z_height_entries)
105        self.geometry("950x" + str(self.window_height))
106
107        def start_entry(self):
108            # Új bejegyzés hozzáadása a Z magasság és relatív srség
beállításához
109
```

```
110     self.label_text = str(1 + len(self.z_height_entries)) + ". Z  
módosítás:"  
111  
112     z_height_label = customtkinter.CTkLabel(self.frame_3, text=self.  
label_text)  
113     z_height_label.grid(row=3 + len(self.z_height_entries), column  
=0, padx=10, pady=10)  
114  
115     z_height_entry = customtkinter.CTkEntry(self.frame_3)  
116     z_height_entry.grid(row=3 + len(self.z_height_entries), column  
=1, padx=0, pady=10)  
117     z_height_entry.insert(0, str(c.s))  
118     z_height_entry.configure(state="disabled")  
119     print(z_height_entry.get())  
120  
121     rel_den_slider = customtkinter.CTkSlider(self.frame_3, from_=45,  
to=100, number_of_steps=55, width=180)  
122     rel_den_slider.grid(row=3 + len(self.rel_den_sliders), column=2,  
padx=10, pady=10)  
123     rel_den_slider.set(100)  
124  
125     rel_den_slider.configure(command=lambda value=rel_den_slider,  
label=self.rel_den_label: self.update_rel_den_label(value, label))  
126  
127     self.z_height_entries.append(z_height_entry)  
128     self.rel_den_sliders.append(rel_den_slider)  
129     self.z_height_labels.append(z_height_label)  
130  
131     self.window_height = 455 + 55 * len(self.z_height_entries)  
132     self.geometry("950x" + str(self.window_height))  
133  
134     def update_rel_den_label(self, value, label):  
135         # Frissíti a c.rel_den értékét és a c.rel_den értékének  
megjelenítését a c.rel_den_label-ben  
136         c.rel_den = value / 100  
137         label.configure(text=("Relatív srség: " + str(value) + "%"))  
138  
139     def remove_entry(self):  
140         for z_height_entry in self.z_height_entries:  
141             z_height_entry.destroy()  
142         self.z_height_entries = []  
143  
144         for rel_den_slider in self.rel_den_sliders:  
145             rel_den_slider.destroy()  
146         self.rel_den_sliders = []  
147  
148         for z_height_label in self.z_height_labels:  
149             z_height_label.destroy()
```

```
150     self.z_height_labels = []
151
152     def mentes(self):
153         if c.method_index == 0:
154             f.mentes_hom(c.rel_den)
155         elif c.method_index == 1:
156             c.hab_top = self.slider1.get()
157             c.hab_base = self.slider2.get()
158             f.mentes_grad(c.hab_base, c.hab_top)
159         elif c.method_index == 2:
160             # G-kód fájl mentése frissített rel_den értékekkel a
megadott Z magasságokon
161             for i in range(len(self.z_height_entries)):
162                 c.z_heights_array.append(float(self.z_height_entries[i].
get()))
163                 c.rel_den_array.append(self.rel_den_sliders[i].get() /
100) # Az érték átalakítása 0 és 1 közötti float típusú
164                 print(c.z_heights_array)
165                 f.mentes_lepcso(c.rel_den_array, c.z_heights_array)
166
167         if c.finish == 1:
168             self.destroy()
169
170     def slider1_event(self, value):
171         c.rel_den = value / 100
172         self.label_slider1.configure(text=("Relatív srség: " + str(value
) + "%"))
173
174     def slider2_event(self, value):
175         c.rel_den = value / 100
176         self.label_slider2.configure(text=("Relatív srség: " + str(value
) + "%"))
177
178     def optionmenu_callback(self, choice):
179         # Régi sliderek eltávolítása
180         if hasattr(self, 'slider1'):
181             self.slider1.destroy()
182             self.label_slider1.destroy()
183
184         if hasattr(self, 'slider2'):
185             self.slider2.destroy()
186             self.label_slider2.destroy()
187
188         if hasattr(self, 'rel_den_label'):
189             self.rel_den_label.destroy()
190
191         if hasattr(self, 'add_entry_button'):
192             self.add_entry_button.destroy()
```

```
193     self.remove_entry()
194
195
196     # Új slider létrehozása
197     if choice == "Homogén habosítás":
198         self.slider1 = customtkinter.CTkSlider(self.frame_3, from_
199 =45, to=100, number_of_steps=55, command=self.slider1_event, width
200 =180)
201
202         self.slider1.grid(row=2, column=0, padx=10, pady=10, sticky=
203 "ew")
204
205         self.slider1.set(100)
206
207         self.label_slider1 = customtkinter.CTkLabel(self.frame_3,
208 text="Relatív srség: 100%")
209         self.label_slider1.grid(row=2, column=1, padx=(0,10))
210         c.method_index = 0
211         self.geometry("750x510")
212
213     elif choice == "Gradiens habosítás":
214         self.slider1 = customtkinter.CTkSlider(self.frame_3, from_
215 =45, to=100, number_of_steps=55, command=self.slider1_event, width
216 =180)
217
218         self.slider1.grid(row=4, column=0, padx=10, pady=10, sticky=
219 "we")
220
221         self.slider1.set(100)
222
223         self.label_slider1 = customtkinter.CTkLabel(self.frame_3,
224 text="Relatív srség: 100%")
225         self.label_slider1.grid(row=2, column=0, padx=(0,10))
226
227         self.slider2 = customtkinter.CTkSlider(self.frame_3, from_
228 =45, to=100, number_of_steps=55, command=self.slider2_event, width
229 =180)
230
231         self.slider2.grid(row=5, column=0, padx=10, pady=10, sticky=
232 "we")
233
234         self.slider2.set(100)
235         self.geometry("750x560")
236
237         self.label_slider2 = customtkinter.CTkLabel(self.frame_3,
238 text="Relatív srség: 100%")
239         self.label_slider2.grid(row=3, column=0, padx=(0,10))
240
241         c.method_index = 1
242
243     elif choice == "Homogén, lépcss habosítás":
244         c.method_index = 2
245
246         self.optionmenu.grid(columnspan=2)
```

```
229
230         self.add_entry_button = customtkinter.CTkButton(master=self.
frame_3, text="Z magasság hozzáadása", command=self.add_entry)
231         self.add_entry_button.grid(row=1, column=2, padx=10, pady
=10, columnspan=1, sticky="we")
232
233         self.rel_den_label = customtkinter.CTkLabel(self.frame_3,
text="Relatív srség: 100%")
234         self.rel_den_label.grid(row=2, column=0, columnspan=3, padx
=10, pady=10)
235
236         self.start_entry()
237     def anyag_menu(self, choise):
238         if choise == "LW-PLA":
239             c.color = c.color_gray
240             c.material = choise
241             img = f.front_view(c.color)
242             self.label1.configure(image = img)
243
244         elif choise == "Variosshore TPU":
245             c.color = c.color_blue
246             c.material = choise
247             img = f.front_view(c.color)
248             self.label1.configure(image = img)
249
250
251 run = App()
252
253 run.mainloop()
```

D. Posztprocesszor, függvények

```
1 import re
2 from PIL import Image, ImageDraw
3 import constants as c
4 import base64
5 import customtkinter
6 import math as m
7 from os import getenv
8
9 def partimage():
10     n = 0
11     e_data = ""
12     for line in c.lines:
13         if 'thumbnail' in line:
14             n+=1
15             continue
16         if n>=2:
17             break
18         if n>0:
19             line = re.sub('; ',',',line)
20             line = re.sub('\n',',',line)
21             e_data += line
22
23     de_data = base64.b64decode(bytes(e_data,'utf-8'))
24     img_file = open(c.os.path.join(c.current_path,'partim.jpg'), 'wb')
25     img_file.write(de_data)
26     img_file.close()
27
28 def front_view(color):
29     with Image.open(c.os.path.join(c.current_path, "bckg.png")) as im:
30         z = 0
31         x_max = 0
32         x_min = 718
33         arany = 718/350
34
35
36         for line in c.lines_tidy:
37
38             if "G1 " in line and "X" in line:
39                 poz_line = re.findall(r'\d+', line)
40                 x = float(poz_line[1])
41                 if x > x_max: x_max = x
42                 if x < x_min: x_min = x
43
44             if "G1 " in line and "E-" in line:
45                 draw = ImageDraw.Draw(im)
46                 if x_max!=0 and x_min!=718:
```

```

47         draw.line((x_min*arany,718-100-(z*arany),x_max*arany
,718-100-(z*arany)),fill = color, width = 1)
48         x_max = 0
49         x_min = 718
50
51         if ";LAYER_CHANGE" in line:
52             draw = ImageDraw.Draw(im)
53             if x_max!=0 and x_min!=718:
54                 draw.line((x_min*arany,718-100-(z*arany),x_max*arany
,718-100-(z*arany)),fill = color, width = 1)
55                 z +=c.s
56                 x_max = 0
57                 x_min = 718
58
59         pic = customtkinter.CTkImage(im,size=(430,430))
60         return pic
61
62 def hToT(hab):
63     if c.material == "LW-PLA":
64         t = PLAhToT(hab)
65         return t
66     elif c.material == "Variosshore TPU":
67         t = TPUhToT(hab)
68         return t
69     else:
70         print("Anyagválasztási hiba")
71
72 def PLAhToT(p):
73     T = round((-1*m.log(-9.91437343755E-31* (0.42-p))-8.55564*c.Q)
/(0.324411 - 0.0395682*c.Q),2)
74     return T
75
76 def TPUhToT(p):
77     T = round(-8.51189*m.log(1.62934E-11 * (-0.5114745805132639 + p))
,2)
78     return T
79
80 def mentes_hom(rel_den):
81     slip_compensation()
82     for i in range(len(c.lines_layer_change)):
83         c.lines_tidy[c.lines_layer_change[i]] = ("M221 S" + str(rel_den
*100) + "\n" + "M220 S" + str(round(1/rel_den*100,2)) + "\n" + "M104
S" + str(round(hToT(rel_den),2)) + "\n" )
84         with open(c.sourceFile,'w') as wf:
85             wf.writelines(c.lines_tidy)
86
87         wf.close()
88         c.finish = 1

```

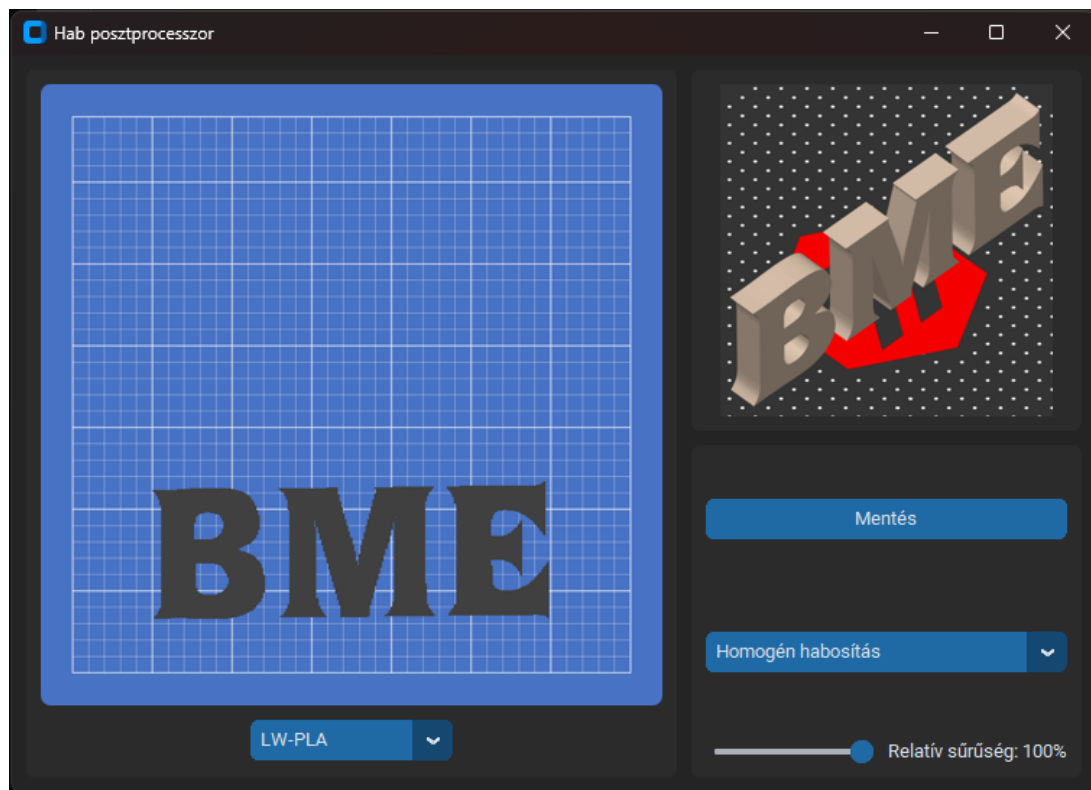
```
89
90 def mentes_grad(hab_min, hab_max):
91     slip_compensation()
92     z_max = round(c.s * len(c.lines_layer_change), 1)
93     for i in range(len(c.lines_layer_change)):
94         c.z_array.append(round(c.s * (i + 1), 1))
95         c.h_array.append(round(hab_min + (hab_max - hab_min) / (len(c.
lines_layer_change) - 1) * (i)))
96         c.t_array.append(hToT(c.h_array[i]))
97
98     for i in range(len(c.lines_layer_change)):
99         c.lines_tidy[c.lines_layer_change[i]] = (("M221 S" + str(100 - c.
h_array[i]) + "\n" + "M220 S" + str(round(100 * 100 / (100 - c.h_array[i])
, 2)) + "\n" + "M104 S" + str(c.t_array[i]) + "\n"))
100
101     with open(c.sourceFile, 'w') as wf:
102         wf.writelines(c.lines_tidy)
103
104     wf.close()
105     c.finish = 1
106
107 def mentes_lepcso(rel_den_array, z_heights):
108     slip_compensation()
109     for i in range(len(c.lines_layer_change)):
110         for j in range(len(z_heights)):
111             if z_heights[j] == c.z_array[i]:
112                 c.lines_tidy[c.lines_layer_change[i]] = ("M221 S" + str(
rel_den_array[j] * 100) + "\n" + "M220 S" + str(round(1 / rel_den_array[j]
* 100, 2)) + "\n" + "M104 S" + str(round(hToT(rel_den_array[j]), 2)) +
"\n" )
113
114     with open(c.sourceFile, 'w') as wf:
115         wf.writelines(c.lines_tidy)
116
117     wf.close()
118     c.finish = 1
119
120 def open_file():
121     with open(c.sourceFile, 'r') as f:
122         for line in f:
123             c.lines.append(line)
124             if "; first_layer_height" in line:
125                 s_array = re.findall(r'\d+', line)
126                 c.s = float(s_array[0] + "." + s_array[1])
127     f.close()
128
129 def tidy_file():
130     cnt = 0
```

```
131     for line in c.lines:
132         if ";LAYER_CHANGE" in line:
133             c.lines_tidy.append(line)
134             c.lines_layer_change.append(cnt)
135             cnt += 1
136         else:
137             c.lines_tidy.append(line)
138             cnt += 1
139     for i in range(len(c.lines_layer_change)):
140         c.z_array.append(round(c.s * (i + 1), 2))
141
142
143 def slip_compensation():
144     if c.material == "LW-PLA":
145         c.slip_const = 1
146     elif c.material == "Varioshore TPU":
147         c.slip_const = 1.050973021
148         print("tpu kiválasztva")
149     else:
150         print("Anyagválasztási hiba")
151     cnt = 0
152     for line in c.lines:
153         if "G1 " in line and "E" in line:
154             line = str(line)
155             extrud_len_str = line[line.find("E")+1:len(line)-1]
156             if "F" not in extrud_len_str and extrud_len_str != "":
157                 extrud_len = float(extrud_len_str)
158                 extrud_len_update = round(extrud_len * c.slip_const, 5) #
159                 extruder kereken történ kúszás kompenzálása
160                 line = line.replace(extrud_len_str, str(extrud_len_update
161 ))
162
163                 c.lines_tidy[cnt] = line
164                 cnt += 1
165             else:
166                 c.lines_tidy[cnt] = line
167                 cnt += 1
168         else:
169             c.lines_tidy[cnt] = line
170             cnt += 1
171
172 def start_routine():
173     open_file()
174     partimage()
175     tidy_file()
176     Q_print = float(getenv('SLIC3R_max_volumetric_speed'))
177     Q_filament = float(getenv('SLIC3R_filament_max_volumetric_speed'))
178     c.Q = min(Q_print, Q_filament)
```

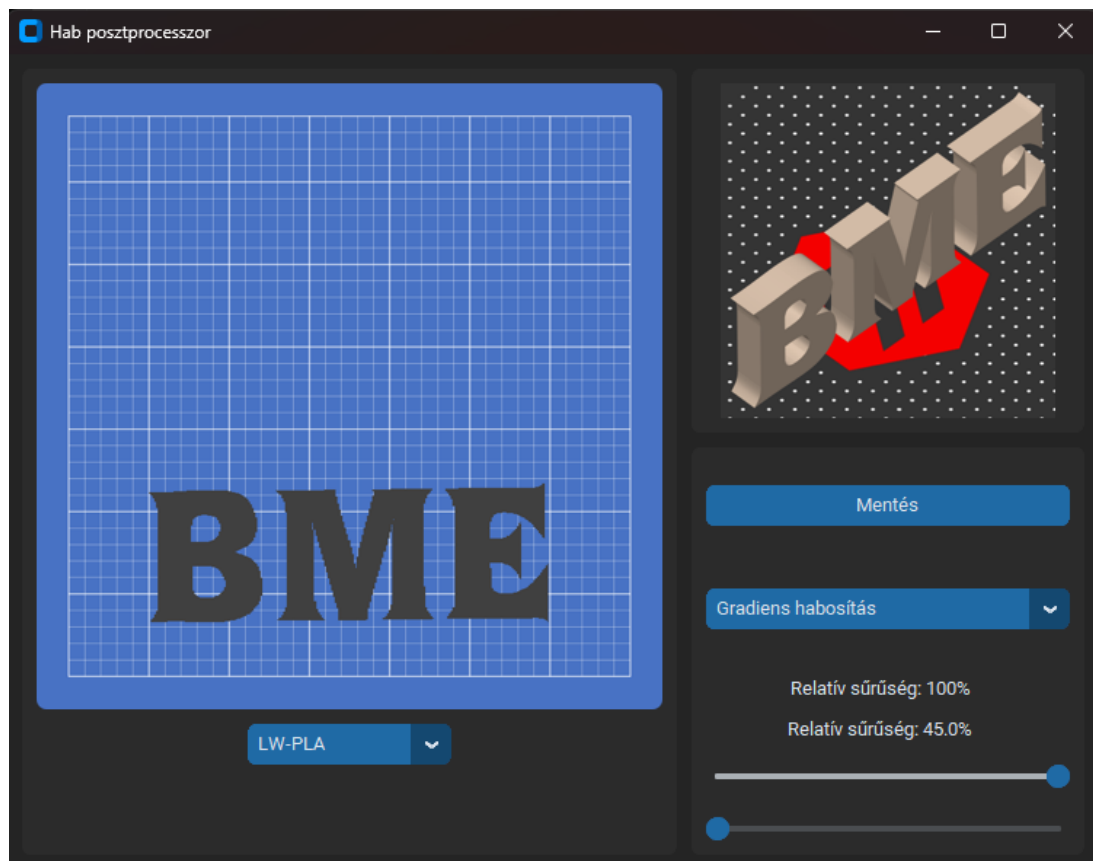
E. Posztprocesszor, konstansok

```
1 import os
2
3 sourceFile = ""
4
5 current_path = os.path.dirname(os.path.realpath(__file__))
6
7 Q = 5
8 rel_den = 1
9 hab_base = 0
10 hab_top = 60
11
12 color_black = (0,0,0)
13 color_gray = (64,64,64)
14 color_blue = (0,0,128)
15 material = "LW-PLA"
16 color = color_gray
17
18 finish = 0
19
20 method_index = 0
21
22 lines = []
23 lines_tidy = []
24 lines_layer_change = []
25
26 s = 0
27
28 z_array = []
29 h_array = []
30 t_array = []
31
32 rel_den_array = []
33 z_heights_array=[]
34 slip_const = 1
```

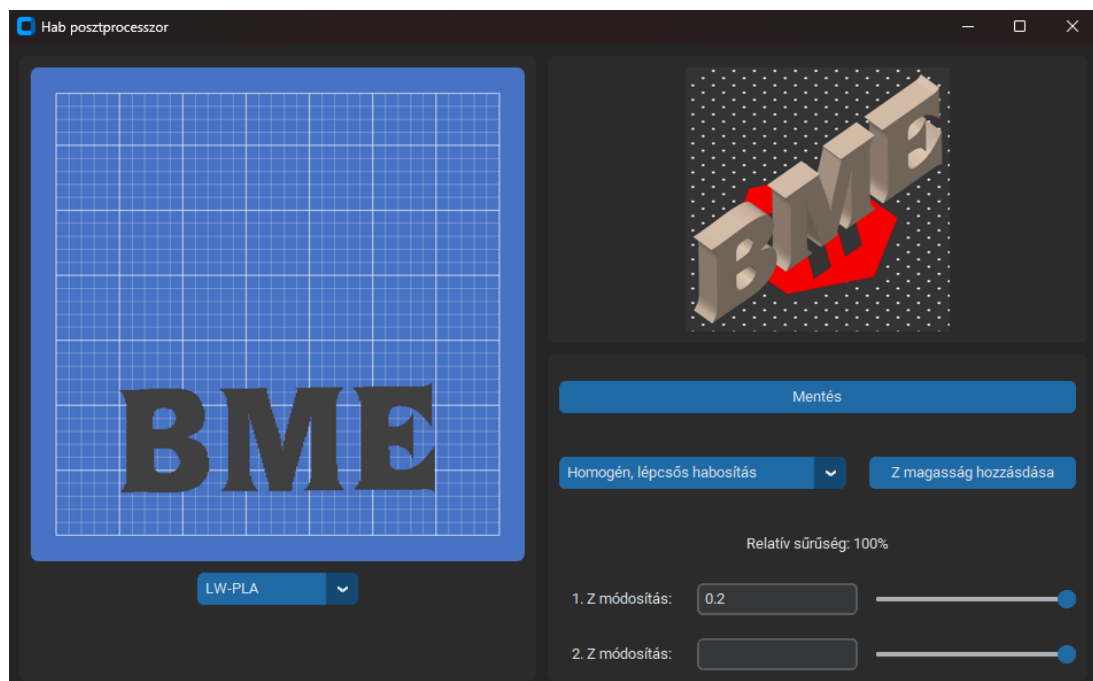
F. Posztprocesszor képek



F.1. ábra. Homogén habosítás opció



F.2. ábra. Gradiens habosítás opció



F.3. ábra. Homogén, lépcsős habosítás opció

G. Mérési adatok feldolgozása

```
1 SetDirectory[NotebookDirectory[]]
2 dir = NotebookDirectory[];
3
4 reldenData = {{{"1", "1", "1.0"}, {"09", "2", "0.9"}, {"08", "
5     "0.8"}, {"07", "4", "0.7"}, {"06", "5", "0.6"}, {"05", "6
6     "0.5"}, {"045", "7", "0.45"}}};
7 dData = {{9.96, 9.95}, {9.92, 9.92}, {9.94, 9.94}, {9.9,
8     9.9}, {9.89,
9     9.89}, {9.85, 9.85}, {9.74, 9.74}}};
10 densityChooser = 2;
11 relden = reldenData[[densityChooser]];
12 linearMaxVal = 1100;
13 filepath = dir <> "\\\" <> relden[[1]];
14 allFiles = FileNames["*.csv", filepath];
15 allData = Drop[Import[#, HeaderLines -> 2], None, 1] & /@
    allFiles;
16 area = \[Pi]/
17     4 Mean[{dData[[densityChooser, 1]]^2,
18     dData[[densityChooser, 2]]^2}];
19 allData[[1]] =
20     Transpose[{allData[[2, All, 1]][[1 ;; -1]],
21     MovingAverage[allData[[2, All, 2]], 1]}];
22 allData[[2]] =
23     Transpose[{allData[[2, All, 1]][[1 ;; -1]],
24     MovingAverage[allData[[2, All, 2]], 1]}];
25 minLen = Min[{Length[allData[[1]]], Length[allData[[2]]]}];
26 dataAvg =
27     1/2 (Take[allData[[1]], minLen] + Take[allData[[2]], minLen
28     ] );
29 Fcondition = # >= linearMaxVal &;
30 slopeFvalue = Select[dataAvg[[All, 2]], Fcondition];
31 Fpos = First[First[Position[dataAvg[[All, 2]], slopeFvalue
32     [[1]]]]];
33 spacing = 3;
34 dataAvg[[Fpos]];
```

```

33 discDiff = (
34   dataAvg[[Fpos + spacing , 2]] - dataAvg[[Fpos - spacing ,
35     2]])/(
36   dataAvg[[Fpos + spacing , 1]] - dataAvg[[Fpos - spacing ,
37     1]]);
38 erinto[x_] := (discDiff (x - dataAvg[[Fpos, 1]]) +
39   dataAvg[[Fpos, 2]]);
40 zerushely = Solve[erinto[x] == 0];
41 dataAvg[[A11, 1]] = dataAvg[[A11, 1]] - zerushely[[1, 1, 2]];
42 dataFiltered = Select[dataAvg, #[[1]] >= 0 &];
43 maxDev = 10;
44 dataFilteredSwapped =
45   Map[{#[[1]],
46     If[#[[2]] < linearMaxVal*0.5,
47       If[Abs[#[[2]] - erinto[#[[1]]]] > maxDev,
48         erinto[#[[1]]], #[[2]], #[[2]]]} &, dataFiltered];
49 \[CurlyEpsilon]t = Log[(30 - dataFilteredSwapped[[A11, 1]])
50   /20];
51 \[Sigma]t =
52   dataFilteredSwapped[[A11, 2]]/area (1 + \[CurlyEpsilon]t);
53 feszVektor =
54   Transpose[{dataFilteredSwapped[[A11, 1]]/20,
55     dataFilteredSwapped[[A11, 2]]/area}];
56 feszVektort = Transpose[{\[CurlyEpsilon]t, \[Sigma]t}];
57 (D[erinto[x], x] 20)/area
58 Export[dir <> "\\filtered\\2023-09-21-LW-PLA-filtered-
59   averaged-" <>
60   relden[[2]] <> "-" <> relden[[1]] <> ".csv",
61   dataFilteredSwapped,
62   "CSV"];
63 Export[dir <>
64   "\\filtered_fesz\\2023-09-21-LW-PLA-filtered-averaged-" <>
65   relden[[2]] <> "-" <> relden[[1]] <> ".csv", feszVektor, "
66   CSV"];
67 Export[dir <>
68   "\\filtered_fesz_t\\2023-09-21-LW-PLA-filtered-averaged-t-" <>
69   relden[[2]] <> "-" <> relden[[1]] <> ".csv", feszVektort,
70   "CSV"];

```

```
65 Show[Plot[erinto[x], {x, 0, 10}, PlotRange -> {0, 5000},  
66   PlotStyle -> {Thin, Blue},  
67   AxesLabel -> {"Elmozdulas [mm]", "Ero [N]"}],  
68 ListPlot[{dataFilteredSwapped},  
69   PlotStyle -> {Red, PointSize[0.003]}]]
```

H. LW-PLA, Gibson modell illesztés

```

1 SetDirectory[NotebookDirectory[]]
2 dataSmall = data[[All, 1 ;; -1 ;; 10]];
3
4 Es = 1080;
5 \[Sigma]ys = 60;
6
7 Ef[\[Rho]_] := ( C1 \[Phi]^2 (\[Rho])^2 + C2 (1 - \[Phi]) \[
   Rho]) Es;
8 \[Sigma]y[\[Rho]_] := (C3 (\[Phi] \[Rho])^(3/2) +
9   C4 (1 - \[Phi]) \[Rho]) \[Sigma]ys;
10 \[CurlyEpsilon]y[\[Rho]_] := \[CurlyEpsilon]D[\[Rho]] (1 - 1/
   d^(
11   1/m));
12 \[CurlyEpsilon]D[\[Rho]_] := 1 - CD \[Rho];
13
14 model2k[\[CurlyEpsilon]_, \[Rho]_] :=
15   Piecewise[{{Ef[\[Rho]] \[CurlyEpsilon], \[CurlyEpsilon] < \
16 \[CurlyEpsilon]y[\[Rho]]}, {(Ef[\[Rho]]*\[CurlyEpsilon]y[\[
   Rho]])/
17   d (\[CurlyEpsilon]D[\[Rho]]/(\[CurlyEpsilon]D[\[Rho]] -
   \
18 \[CurlyEpsilon]))^m, \[CurlyEpsilon]y[\[Rho]] < \[
   CurlyEpsilon]}}]
19
20 QNSSRD[list_, function_] :=
21   Sum[errorFunctionNSSRD[list[[i]], function, densities[[i]],
   {i, 1,
22   Length[list]}]
23 errorFunctionNSSRD[list_, function_, \[Rho]_] :=
24   1/Length[
25   list[[All, 1]]] Total[(
26   list[[All, 2]] - (function[#, \[Rho]] & /@ list[[All,
   1]]))/
27   list[[All, 2]]^2]
28
29 paramNSSRD2 =
30   NMinimize[{QNSSRD[dataSmall, model2k], 0.6 < m < 1.4, 0 < CD
   < 0.6,

```

```
31 0.5 < d < 2, 0 < C1, 0 < C2, 0.5 < \[Phi] < 1}, {C1, C2,
32 CD, d,
33 m, \[Phi]}, MaxIterations -> 20, Method -> "RandomSearch"]
34 QNSSD[list_, function_] :=
35 Sum[errorFunctionNSSD[list[[i]], function, densities[[i]]],
36 {i, 1,
37 Length[list]}]
38 errorFunctionNSSD[list_, function_, \[Rho]_] :=
39 1/Length[
40 list[[All,
41 1]]] Total[(list[[All,
42 2]] - (function[#, \[Rho]] & /@ list[[All, 1]]))^2]
43 paramNSSD2 =
44 NMinimize[{QNSSD[dataSmall, model2k], 0.6 < m < 1.4, 0 < CD
45 < 0.6,
46 0.5 < d < 2, 0 < C1, 0 < C2, 0.5 < \[Phi] < 1}, {C1, C2,
CD, d,
m, \[Phi]}, MaxIterations -> 20, Method -> "RandomSearch"]
```

I. LW-PLA, Liu modell illesztés

```
1 FSZ = 10; FFAM = "Cambria";
2 cm = 72/2.54;
3 mm = cm/10.0;
4
5 rawDataChooser = {"LW-PLA-D15-L30", "LW-PLA-D10-L20",
6   "Varioshore-TPU-D15-L30"};
7 densities = {1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.45};
8
9 SetDirectory["../../.."];
10 SetDirectory[
11   "Nyomomeresek\\" <> rawDataChooser[[2]] <>
12   "\\filtered_fesz"];
13 dataFiles = FileNames["*.csv"];
14 data = Import[#] & /@ dataFiles;
15 SetDirectory[NotebookDirectory[]];
16
17 feszlp[data_, label1_, label2_] :=
18   ListPlot[data, LabelStyle -> {FSZ, FontFamily -> FFAM, Black
19     },
20     FrameLabel -> {label1, label2}, Frame -> True, ImageSize ->
21       150 mm,
22     GridLines -> Automatic, PlotStyle -> {Blue, PointSize
23       [0.004]},
24     PlotRange -> {{0, 1}, {0, 60}}]
25 feszllp[data_, label1_, label2_] :=
26   ListLinePlot[data, LabelStyle -> {FSZ, FontFamily -> FFAM,
27     Black},
28     FrameLabel -> {label1, label2}, Frame -> True, ImageSize ->
29       150 mm,
30     GridLines -> Automatic, PlotStyle -> {Blue, Thickness
31       [0.001]},
32     PlotRange -> {{0, 1}, {0, 60}}]
33 feszplt[function_, label1_, label2_] :=
34   Plot[function, {x, 0, 1},
35     LabelStyle -> {FSZ, FontFamily -> FFAM, Black},
36     FrameLabel -> {label1, label2}, Frame -> True, ImageSize ->
37       150 mm,
38     GridLines -> Automatic, PlotStyle -> {Black, Thickness
```

```

    [0.002]],
32   PlotRange -> {All, {0, 60}}]
33 plt[function_, label1_, label2_] :=
34   Plot[function, {x, 0, 1},
35     LabelStyle -> {FSZ, FontFamily -> FFAM, Black},
36     FrameLabel -> {label1, label2}, Frame -> True, ImageSize ->
      150 mm,
37     GridLines -> Automatic, PlotStyle -> {Black, Thickness
      [0.002]],
38     PlotRange -> {All, All}]
39
40 model[\[CurlyEpsilon]_, \[Rho]_] :=
41   A[\[Rho]]*(Exp[\[Alpha][\[Rho]] \[CurlyEpsilon]] - 1)/(
42     B[\[Rho]] + Exp[\[Beta][\[Rho]] \[CurlyEpsilon]]) +
43     Exp[c[\[Rho]]] (Exp[\[Gamma][\[Rho]] \[CurlyEpsilon]] - 1)
44
45 A[\[Rho]_] := k \[Rho]^m
46 B[\[Rho]_] := 1
47 \[Alpha][\[Rho]_] := a\[Alpha]/(1 + b\[Alpha] \[Rho]^c\[Alpha]
    ])
48 \[Beta][\[Rho]_] := a\[Beta]/(1 + b\[Beta] \[Rho]^c\[Beta])
49 c[\[Rho]_] := k1 + k2 \[Rho]
50 \[Gamma][\[Rho]_] := n (1 - \[Rho]/\[Rho]max)
51 \[Rho]max = 0.9
52
53 Q[list_, function_] :=
54   Sum[errorFunction[list[[i]], function, densities[[i]]], {i,
      1,
55     Length[list]}]
56 errorFunction[list_, function_, \[Rho]_] :=
57   1/Length[
58     list] Total[((
59     list[[All, 2]] - (function[#, \[Rho]] & /@ list[[All,
      1]])))/
60     list[[All, 2]])^2]
61
62 dataSmall = data[[All, 1 ;; -1 ;; 10]];
63
64 param1 =
65   NMinimize[{Q[dataSmall, model]}, {k, m, a\[Alpha], b\[Alpha]

```

```

],
66   c\[Alpha], a\[Beta], b\[Beta], c\[Beta], k1, k2, n},
67   Method -> "RandomSearch", MaxIterations -> 1000]
68
69 QNSSD[list_, function_] :=
70   Sum[errorFunctionNSSD[list[[i]], function, densities[[i]]],
71     {i, 1,
72       Length[list]}}
73 errorFunctionNSSD[list_, function_, \[Rho]_] :=
74   1/Length[
75     list] Total[(list[[All,
76       2]] - (function[#, \[Rho]] & /@ list[[All, 1]]))^2]
77 paramNSSD =
78   NMinimize[{QNSSD[dataSmall, model]}, {k, m, a\[Alpha], b\[
79     Alpha],
80     c\[Alpha], a\[Beta], b\[Beta], c\[Beta], k1, k2, n},
81     Method -> "RandomSearch", MaxIterations -> 1000]
82 rsquared[list_, function_, i_] :=
83   1 - SquaredEuclideanDistance[list[[i, All, 2]],
84     function[#, densities[[i]]] /. param1[[2]] & /@
85     list[[i, All, 1]]]/
86     SquaredEuclideanDistance[list[[i, ;; , 2]],
87     Mean@ list[[i, ;; , 2]]];

```

J. Optimális relatív sűrűség kiválasztása

```

1 LiuPLA[\[CurlyEpsilon]_, \[Rho]_] :=
2   A1[\[Rho]]*(Exp\[Alpha]1[\[Rho]] \[CurlyEpsilon]] - 1)/(
3     B1[\[Rho]] + Exp\[Beta]1[\[Rho]] \[CurlyEpsilon])) +
4     Exp[c1[\[Rho]]] (Exp\[Gamma]1[\[Rho]] \[CurlyEpsilon]] -
5       1)
6
7 FSZ = 10; FFAM = "Cambria";
8 cm = 72/2.54;
9 mm = cm/10.0;
10
11 \[Sigma]max1 = 55;
12 \[Sigma]max2 = 50;
13 \[Sigma]max3 = 45;
14 \[Sigma]max4 = 40;
15 \[Sigma]max5 = 30;
16
17 S50 = Plot[{NIntegrate[
18   LiuPLA[\[Epsilon], \[Rho]]/\[Sigma]max1, {\[Epsilon], 0,
19     x /. FindRoot[LiuPLA[x, \[Rho]] - \[Sigma]max1, {x,
20       0}]]],
21   NIntegrate[
22     LiuPLA[\[Epsilon], \[Rho]]/\[Sigma]max2, {\[Epsilon], 0,
23       x /. FindRoot[LiuPLA[x, \[Rho]] - \[Sigma]max2, {x,
24         0}]]],
25     NIntegrate[
26       LiuPLA[\[Epsilon], \[Rho]]/\[Sigma]max3, {\[Epsilon], 0,
27         x /. FindRoot[LiuPLA[x, \[Rho]] - \[Sigma]max3, {x,
28           0}]]],
29       NIntegrate[
30         LiuPLA[\[Epsilon], \[Rho]]/\[Sigma]max4, {\[Epsilon], 0,
31           x /. FindRoot[LiuPLA[x, \[Rho]] - \[Sigma]max4, {x,
32             0}]]],
33         NIntegrate[
34           LiuPLA[\[Epsilon], \[Rho]]/\[Sigma]max5, {\[Epsilon], 0,
35             x /. FindRoot[
36               LiuPLA[x, \[Rho]] - \[Sigma]max5, {x, 0}]]]], {\[Rho],
37             0.4, 1},
38   Frame -> True, GridLines -> Automatic,
39   LabelStyle -> {FSZ, FontFamily -> FFAM, Black},

```

```
33   FrameLabel -> {"Relativ \
34   suruseg, \! \(\(*FractionBox[\(\(*SuperscriptBox[\(\([Rho]\),
      \(*\)]\(\(\ \
35   \)\), SubscriptBox[\(\([Rho]\), \(\s\)]\)\)[1]",
36   "Hatekonysag, \[Eta][\! \(\(*FractionBox[\(Nm\), \(\MPa\)\
      ]\)]"},
37   ImageSize -> 110 mm, FrameStyle -> Black,
38   PlotStyle -> {Black, Red, Blue, Orange, Brown},
39   PlotStyle -> {Black, Red, Blue, Orange, Brown},
40   PlotRange -> {0, 0.35}]
```

K. Poisson tényező - motion tracking

```
1 SetDirectory[NotebookDirectory[]]
2 FSZ = 10; FFAM = "Cambria";
3 cm = 72/2.54;
4 mm = cm/10.0;
5
6 relden = "05";
7 d0 = 15;
8 filepath =
9   SetDirectory[NotebookDirectory[]] <> "\\\" <> relden <>
10   "-tracker-data\\";
11 rootfolder = SetDirectory[NotebookDirectory[]];
12 allFiles = FileNames["*.csv", filepath];
13 allData =
14   Drop[Import[#, HeaderLines -> 2], None, 1 ;; 2] & /@
15   allFiles;
16 \[Epsilon]Original =
17   Import[rootfolder <> "\\2023-09-23-epsilon1-reldensity" <>
18   relden <>
19   ".csv"];
20 filteredData = allData[[All, 1 ;; -1 ;; 5]];
21 dL = Sqrt[(filteredData[[1, All, 2]] -
22   filteredData[[1, All, 1]])^2 + (filteredData[[2, All,
23   2]] -
24   filteredData[[2, All, 1]])^2];
25 dU = Sqrt[(filteredData[[3, All, 2]] -
26   filteredData[[3, All, 1]])^2 + (filteredData[[4, All,
27   2]] -
28   filteredData[[4, All, 1]])^2];
29 d = MovingAverage[(dL + dU)/2, 5];
30 \[Epsilon]OriginalF = x \[Epsilon]Original[[-1]] /Length[d];
31 \[Epsilon]1 = Table\[Epsilon]OriginalF, {x, 1, Length[d],
32   1}];
33 \[Lambda] = 1 - \[Epsilon]1;
34 \[Lambda]T = d/d0;
35 poisson =
36   Transpose[{\[Epsilon]1[[All, 1]],
37     Flatten[-(Log\[Lambda]T)/Log\[Lambda]]}],
38   Export[rootfolder <> "\\2023-09-21-poisson-relden" <> relden
```

```

34     "<>".csv", poisson, "CSV"];
35 ListPlot[poisson,
36   FrameLabel -> {"\!\(\(*SubscriptBox[\(\[CurlyEpsilon]\),
    \(\(1\)\)\)",
37     "\!\(\(*SubscriptBox[\(\[Nu]\), \(\(1, 2\)\)\)\)\!\(\(*
    SubscriptBox[\(\[
38 \[CurlyEpsilon]\), \(\(1\)\)\)\!\(\(*SuperscriptBox[\(\)\), \
39 \(\(Hencky\)\)\)"}], Frame -> True, ImageSize -> {640, 480},
40   GridLines -> Automatic, PlotStyle -> {Red, PointSize
    [0.004]}],
41   PlotRange -> {{0, Automatic}, {0, 1}},
42   PlotLabel ->
43     "Variosshore TPU,\!\(\(*FractionBox[\(\(\(\)\)\)\)\(*
    SuperscriptBox[\(\[
44 \[Rho]\), \(\(*\)\)\), SubscriptBox[\(\[Rho]\), \(\(s\)\)]\)=0." <>
45     StringTake[relden, {2, -1}] <> ", Poisson Function",
46   LabelStyle -> {Bold, Black}]

```

L. Poisson tényező - binarizálás

```
1 SetDirectory[NotebookDirectory[]]
2 allFiles = FileNames["*.mp4"];
3 indexes = {{1, "10"}, {2, "11"}, {3, "12"}, {4, "13"}, {5, "
4     "14"}, {6,
5     "15"}, {7, "17"}};
6 index = 1
7 videos = Video[allFiles[[indexes[[index, 1]]]]
8 hossz10 = QuantityMagnitude[Floor[10*Duration[videos]]]
9 f = Table[VideoExtractFrames[videos, i/10], {i, 0, hossz10}];
10 bin = Table[
11     Binarize[f[[i]], Method -> "MinimumError"], {i, 1, hossz10
12     + 1}];
13 Aw = Table[ImageMeasurements[bin[[i]], "Mean"], {i, 1,
14     hossz10 + 1}];
15 A = Table[
16     ImageDimensions[bin[[i]]][[1]]*ImageDimensions[bin[[i
17     ]]][[2]], {i,
18     1, hossz10 + 1}];
19 d = Table[
20     Transpose[{(i - 1),
21         A[[i]]/ImageDimensions[bin[[i]]][[2]] (1 - Aw[[i]])
22         *0.05}], {i, 1,
23         hossz10 + 1}];
24 Export["poisson_binarizalas" <> indexes[[index, 2]] <>
25     ".csv", d, "CSV"]
```