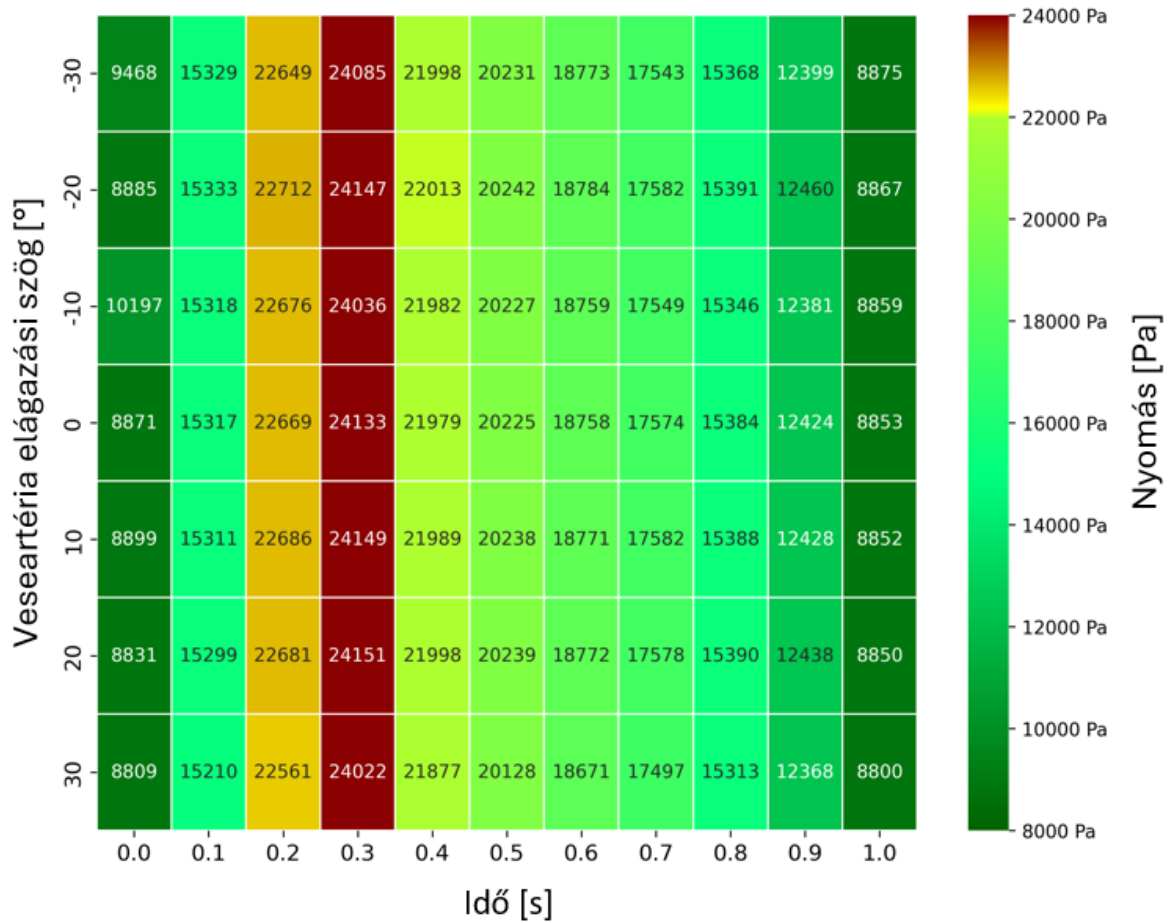


A. függelék

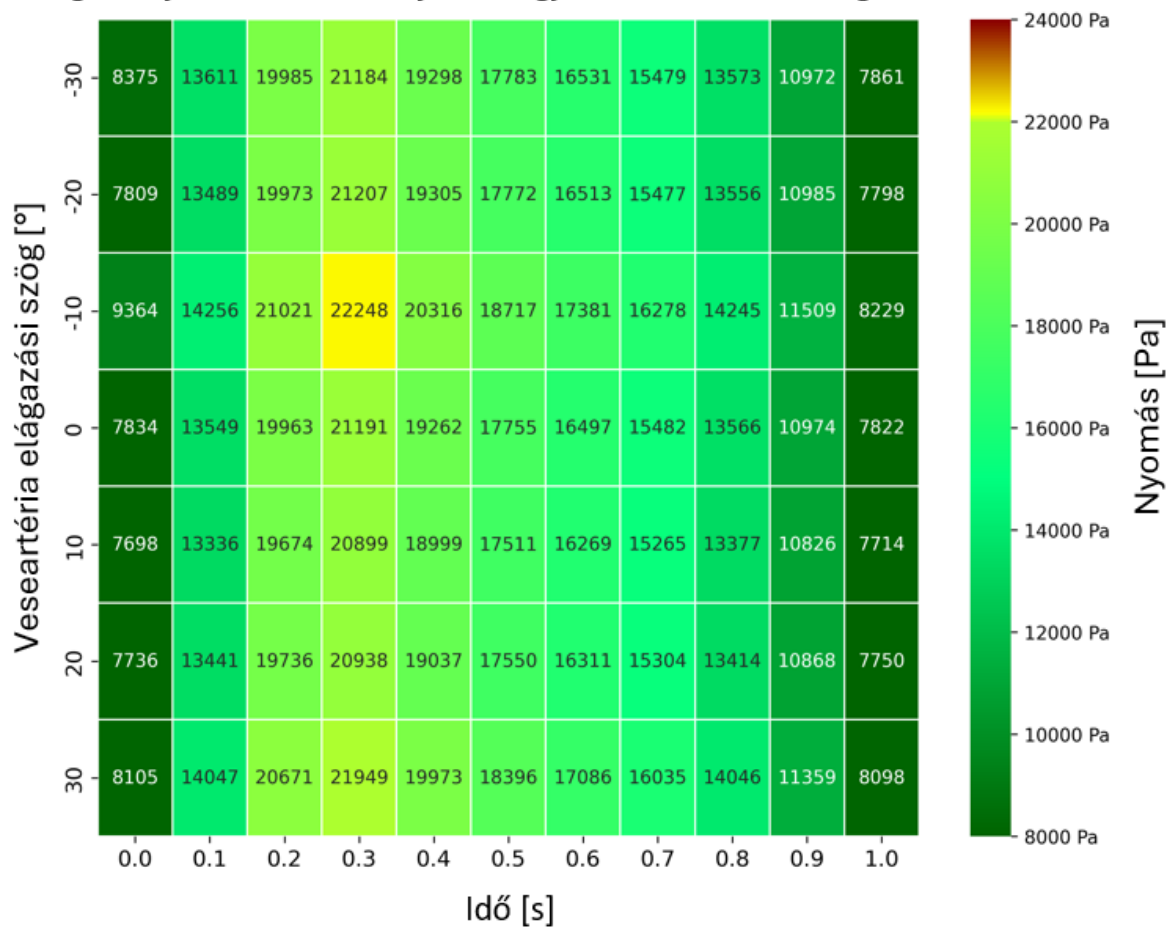
A.1. Modell 1 nyomás értékei

Térfogat súlyozott maximum nyomás egy szív ciklus alatt – szegmens 1



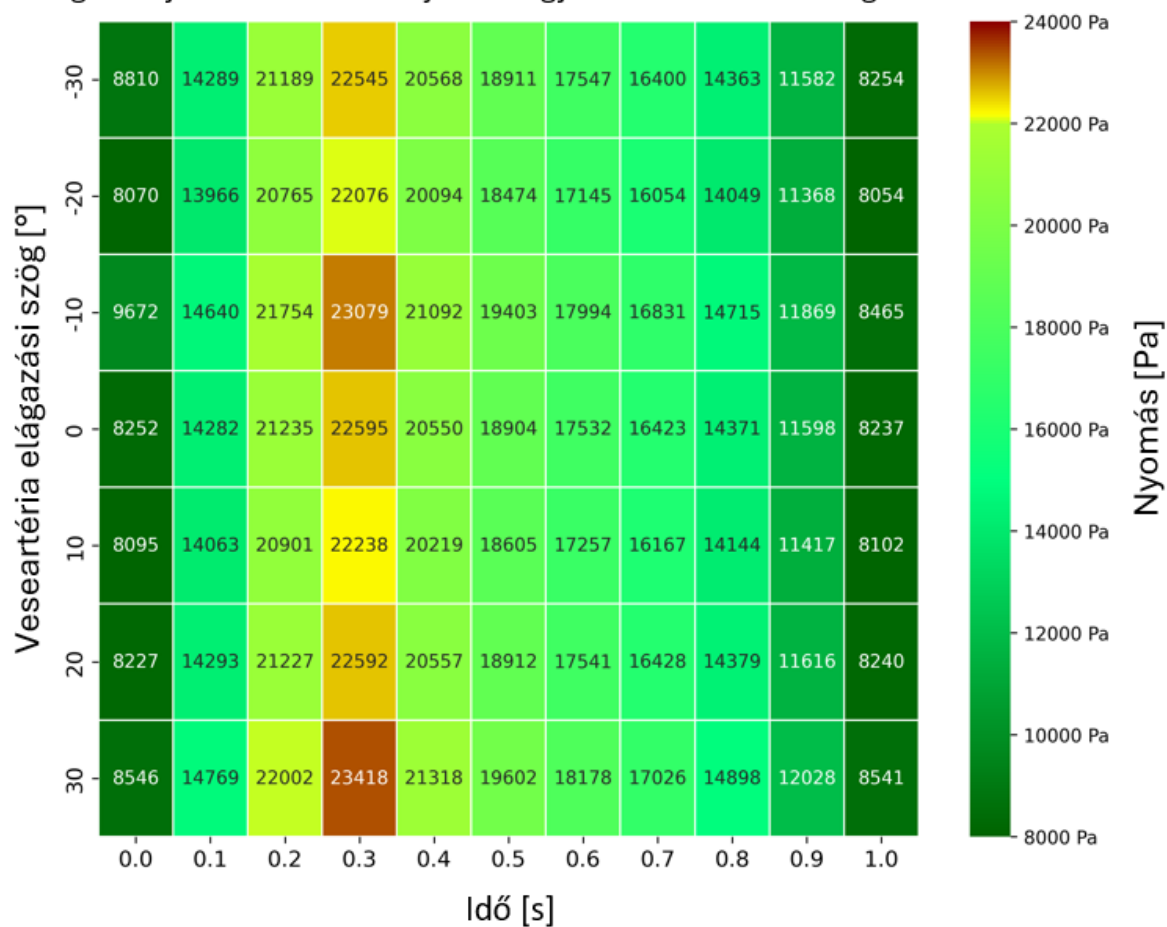
A.1. ábra. Térfogat súlyozott maximális nyomás az 1. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli (8000–22000 Pa), míg a piros a megemelkedett, krónikus szinteket mutatja.

Térfogat súlyozott minimum nyomás egy szívciklus alatt – szegmens 1

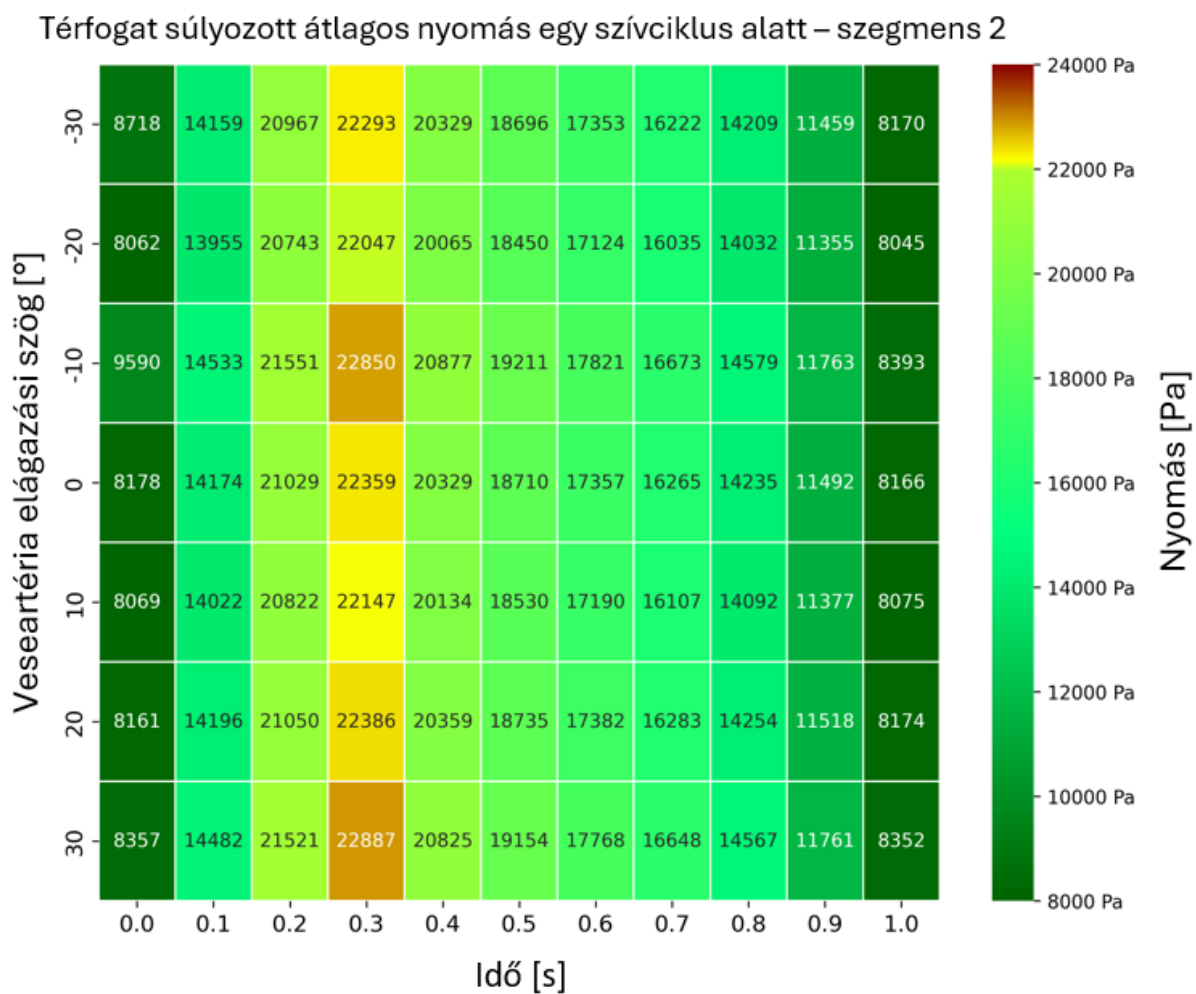


A.2. ábra. Térfogat súlyozott minimális nyomás az 1. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli (8000–22000 Pa), míg a piros a megemelkedett, krónikus szinteket mutatja.

Térfogat súlyozott maximum nyomás egy szívciklus alatt – szegmens 2

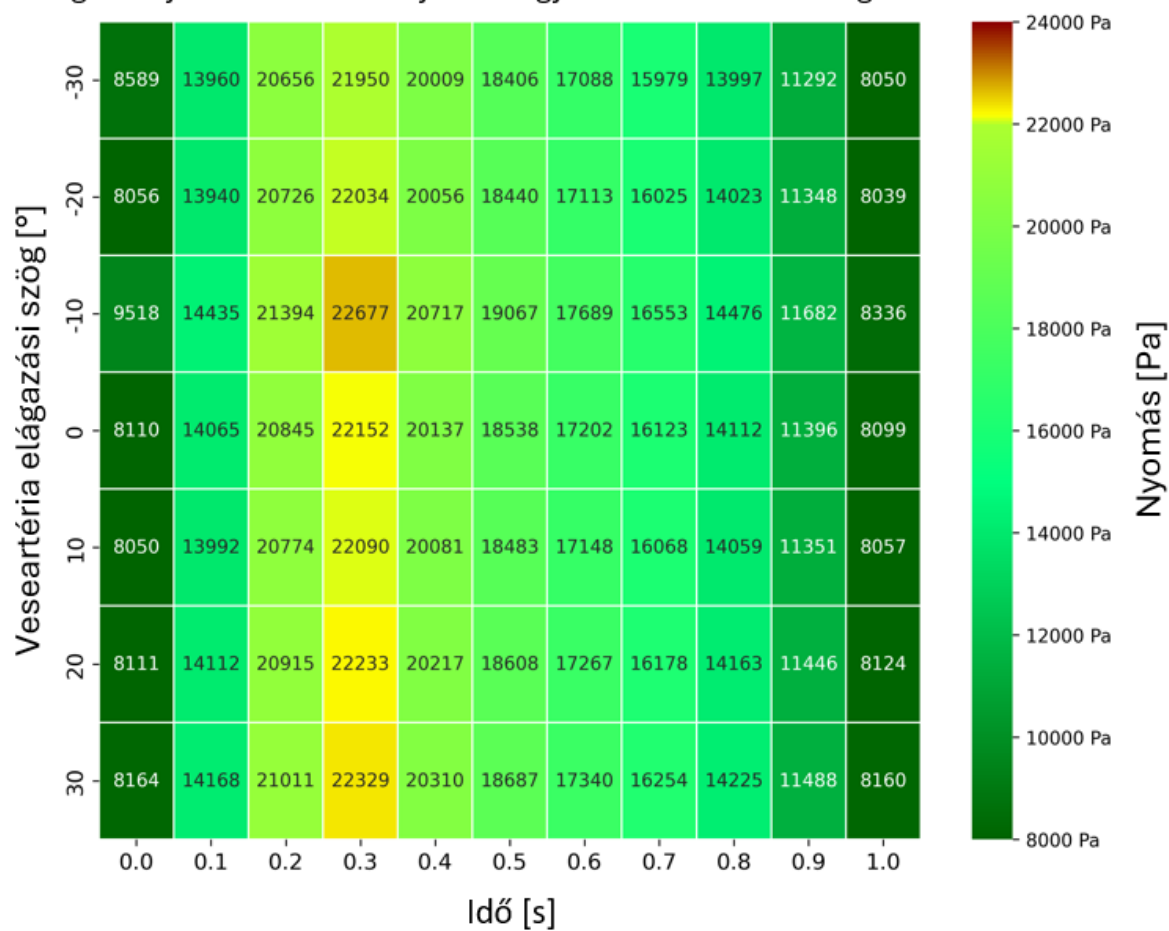


A.3. ábra. Térfogat súlyozott maximális nyomás a 2. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli (8000–22000 Pa), míg a piros a megemelkedett, krónikus szinteket mutatja.



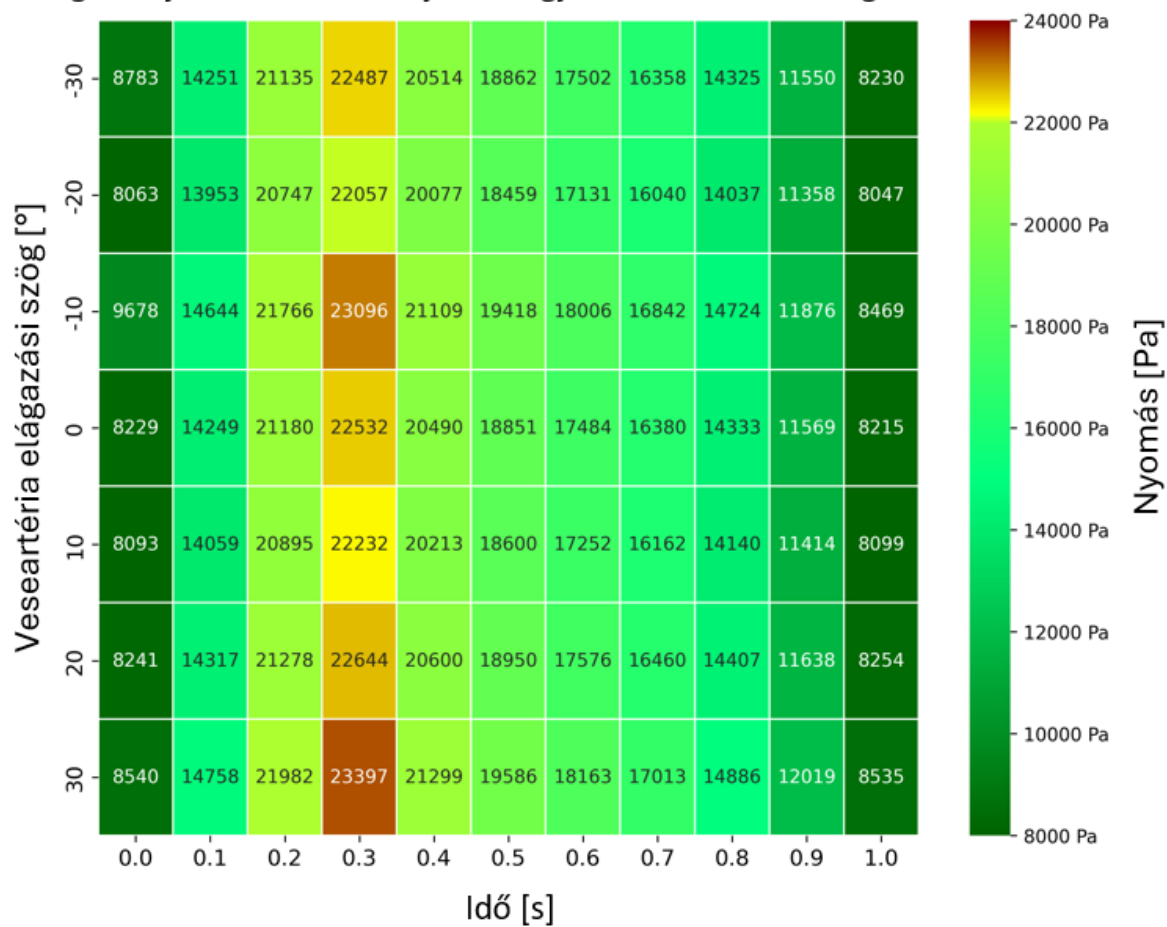
A.4. ábra. Térfogat súlyozott átlagos nyomás a 2. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli (8000–22000 Pa), míg a piros a megemelkedett, krónikus szinteket mutatja.

Térfogat súlyozott minimum nyomás egy szív ciklus alatt – szegmens 2

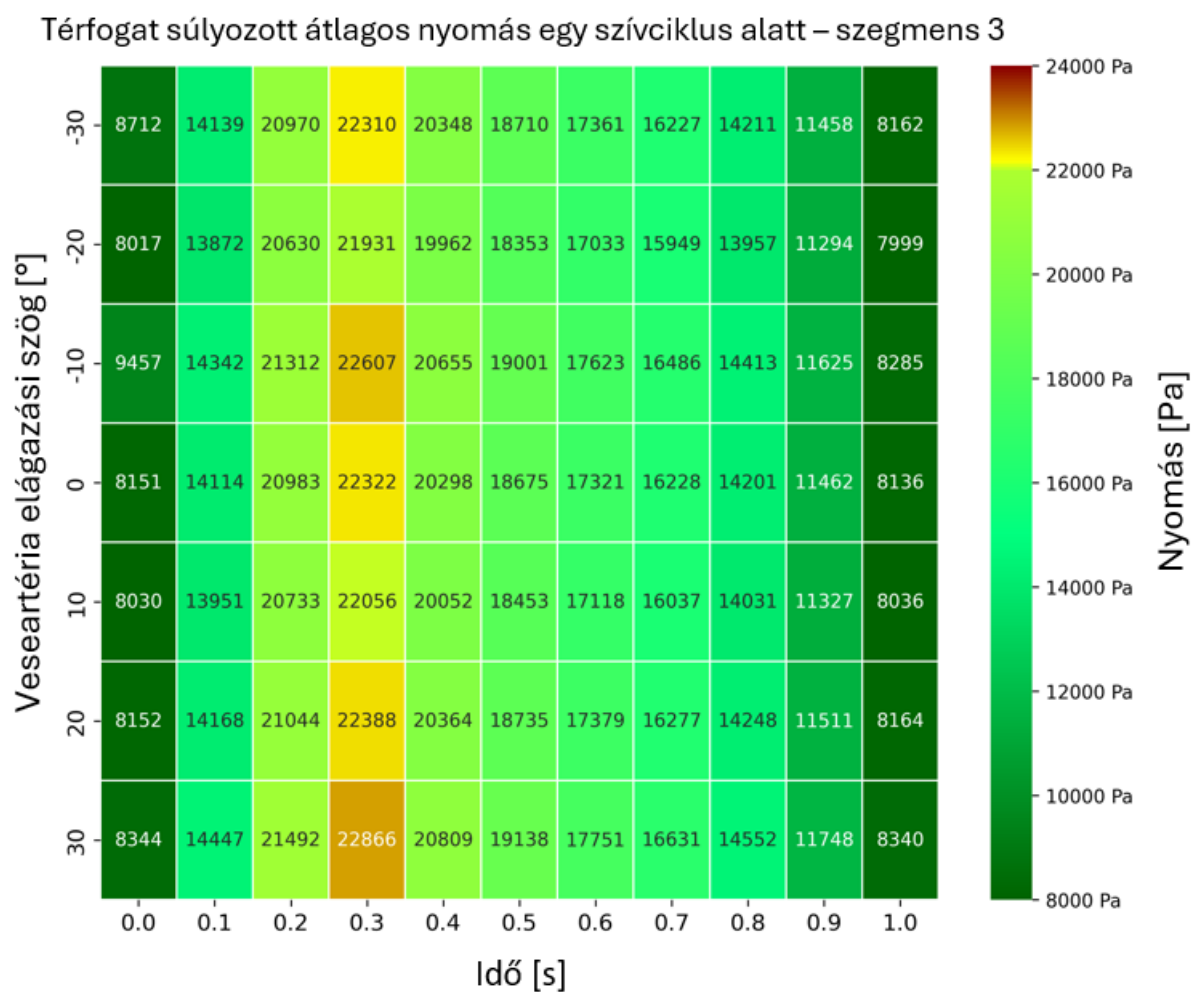


A.5. ábra. Térfogat súlyozott minimális nyomás a 2. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli (8000–22000 Pa), míg a piros a megemelkedett, krónikus szinteket mutatja.

Térfogat súlyozott maximum nyomás egy szívciklus alatt – szegmens 3

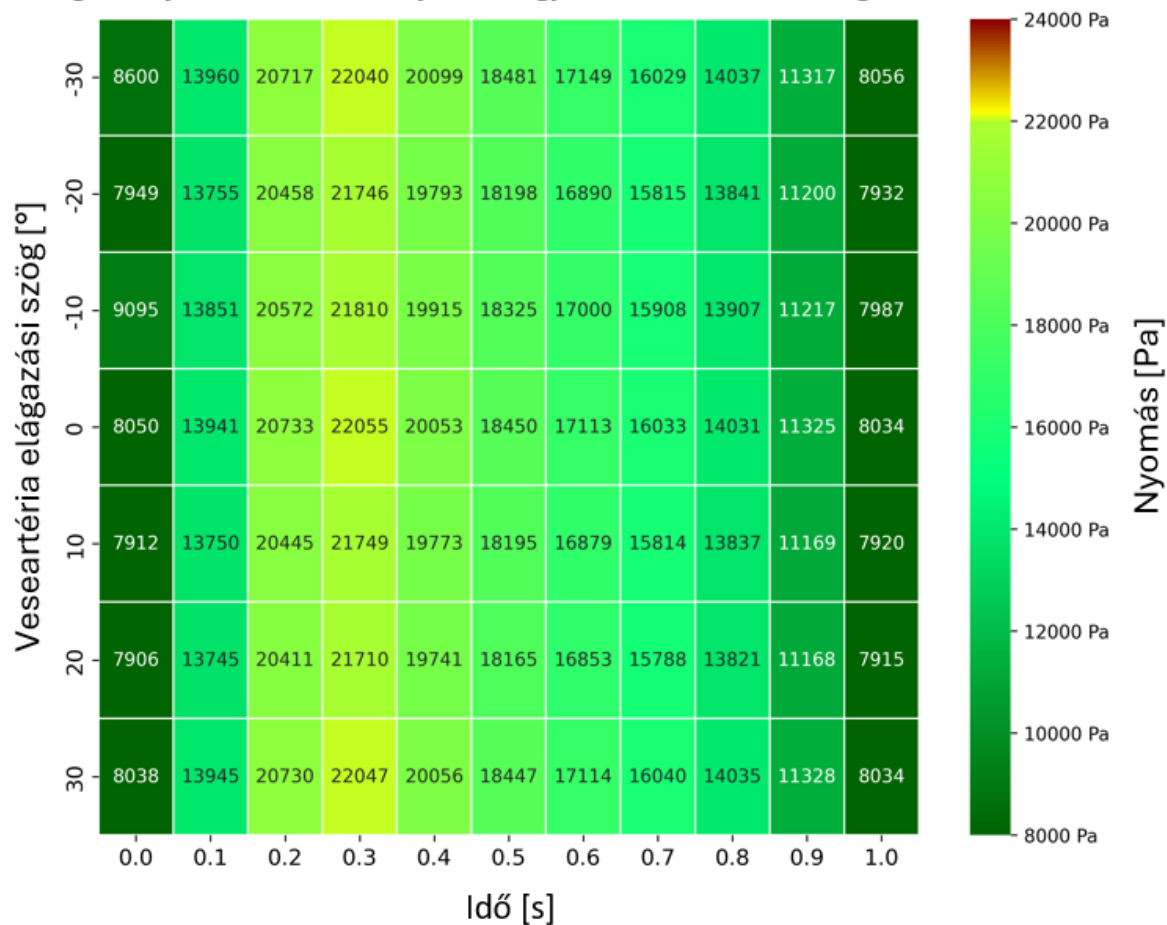


A.6. ábra. Térfogat súlyozott maximális nyomás a 3. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli (8000–22000 Pa), míg a piros a megemelkedett, krónikus szinteket mutatja.



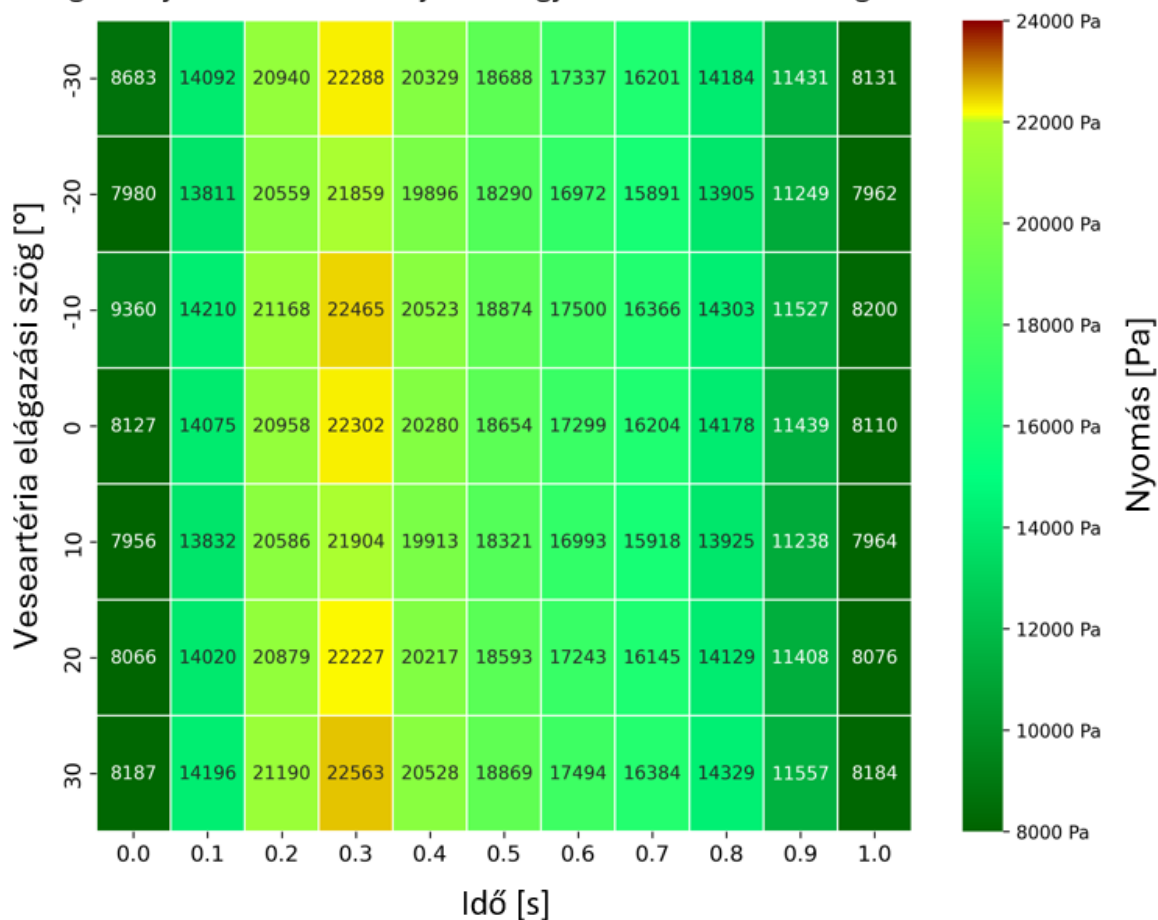
A.7. ábra. Térfogat súlyozott átlagos nyomás a 3. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli (8000–22000 Pa), míg a piros a megemelkedett, krónikus szinteket mutatja.

Térfogat súlyozott minimum nyomás egy szívciklus alatt – szegmens 3

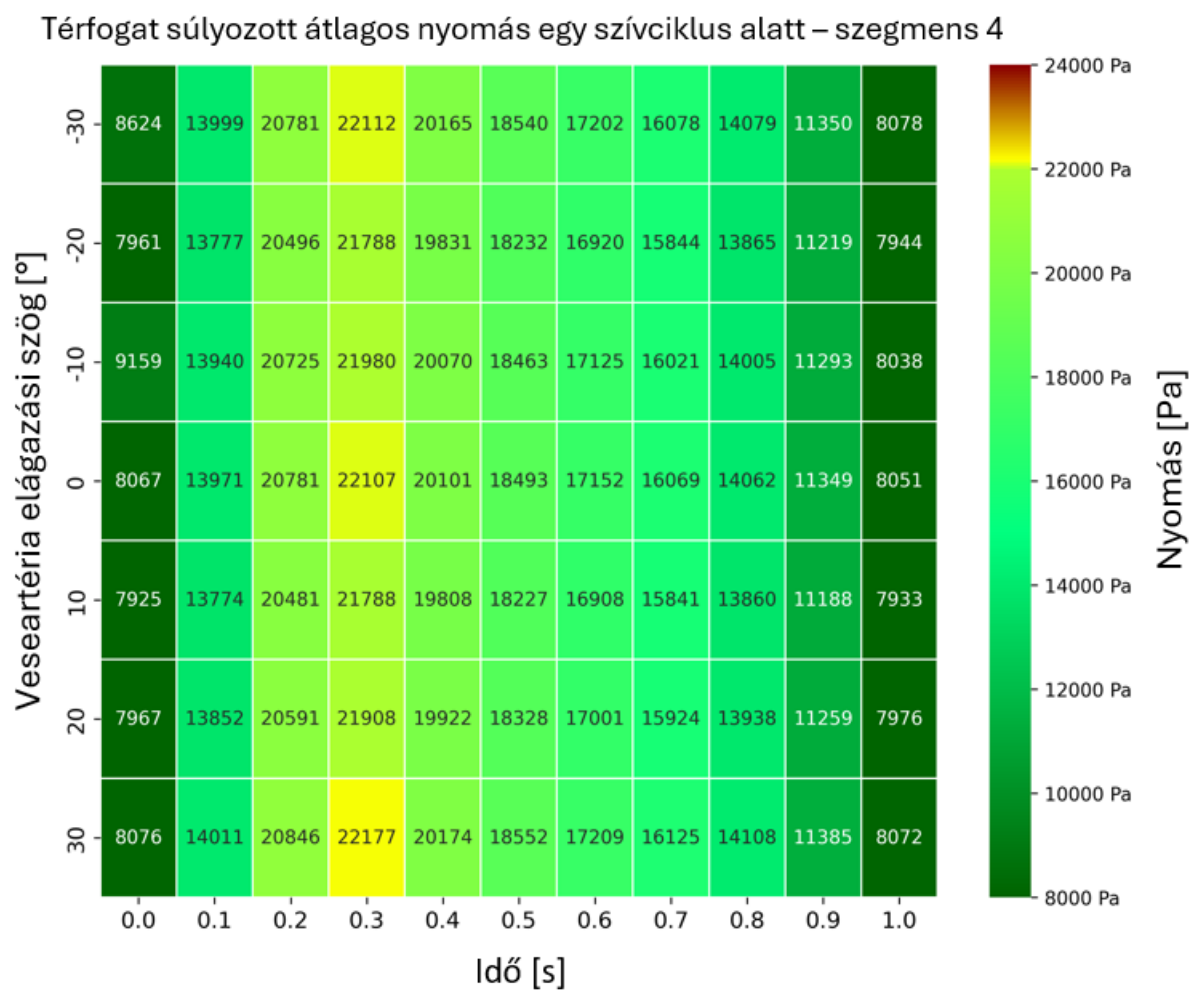


A.8. ábra. Térfogat súlyozott minimális nyomás a 3. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli (8000–22000 Pa), míg a piros a megemelkedett, krónikus szinteket mutatja.

Térfogat súlyozott maximum nyomás egy szívciklus alatt – szegmens 4

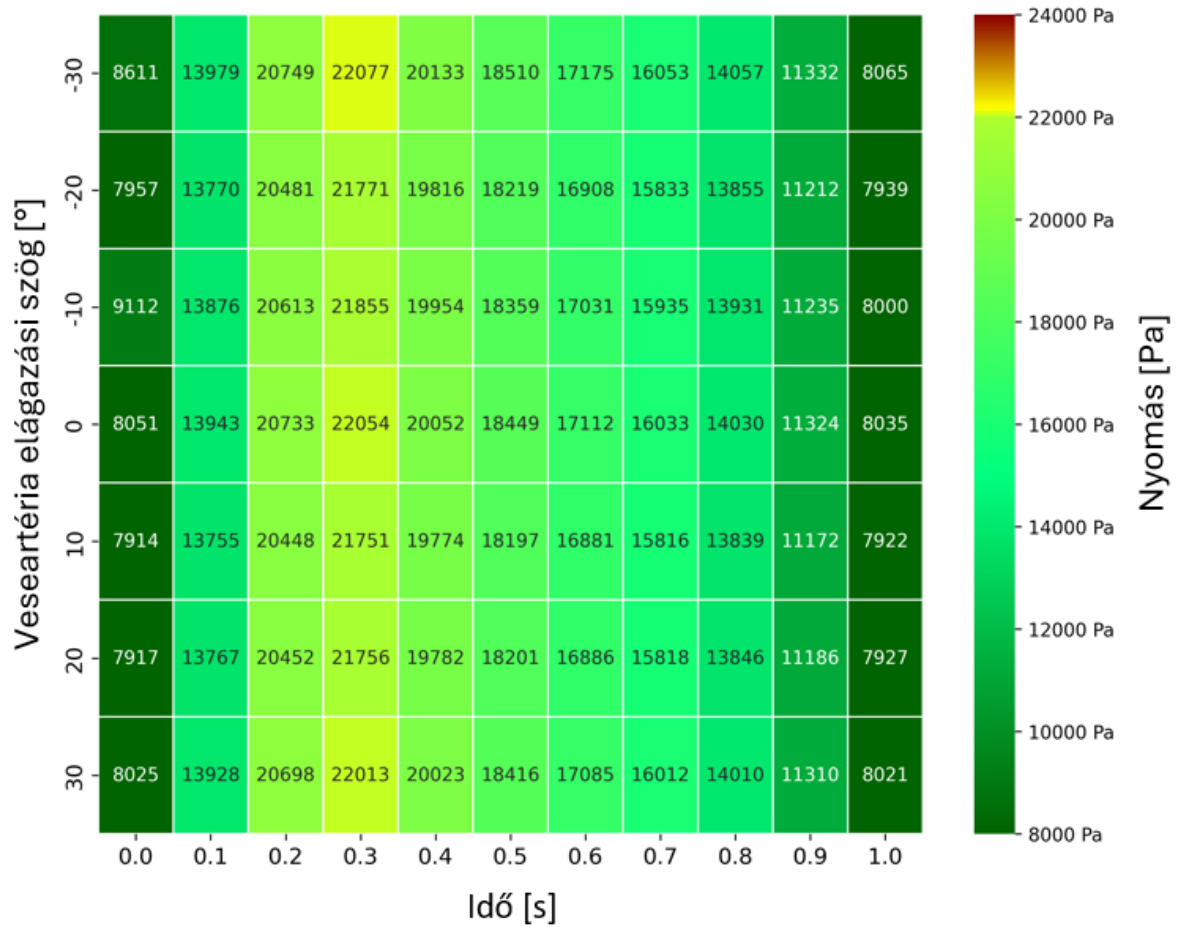


A.9. ábra. Térfogat súlyozott maximális nyomás a 4. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli (8000–22000 Pa), míg a piros a megemelkedett, krónikus szinteket mutatja.



A.10. ábra. Térfogat súlyozott átlagos nyomás a 4. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli (8000–22000 Pa), míg a piros a megemelkedett, krónikus szinteket mutatja.

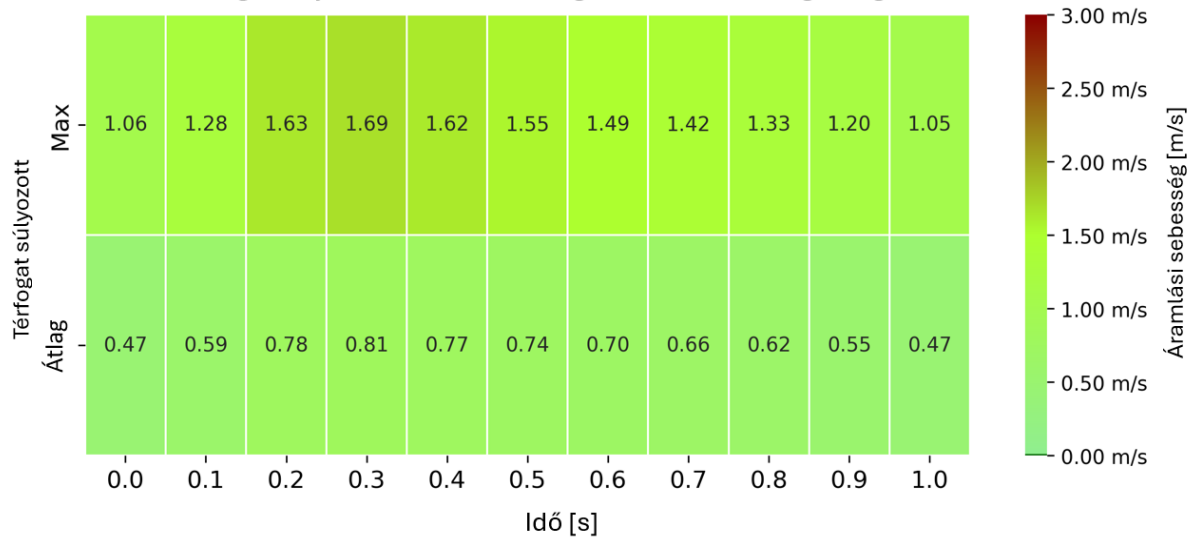
Térfogat súlyozott minimum nyomás egy szívciklus alatt – szegmens 4



A.11. ábra. Térfogat súlyozott minimális nyomás a 4. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli (8000–22000 Pa), míg a piros a megemelkedett, krónikus szinteket mutatja.

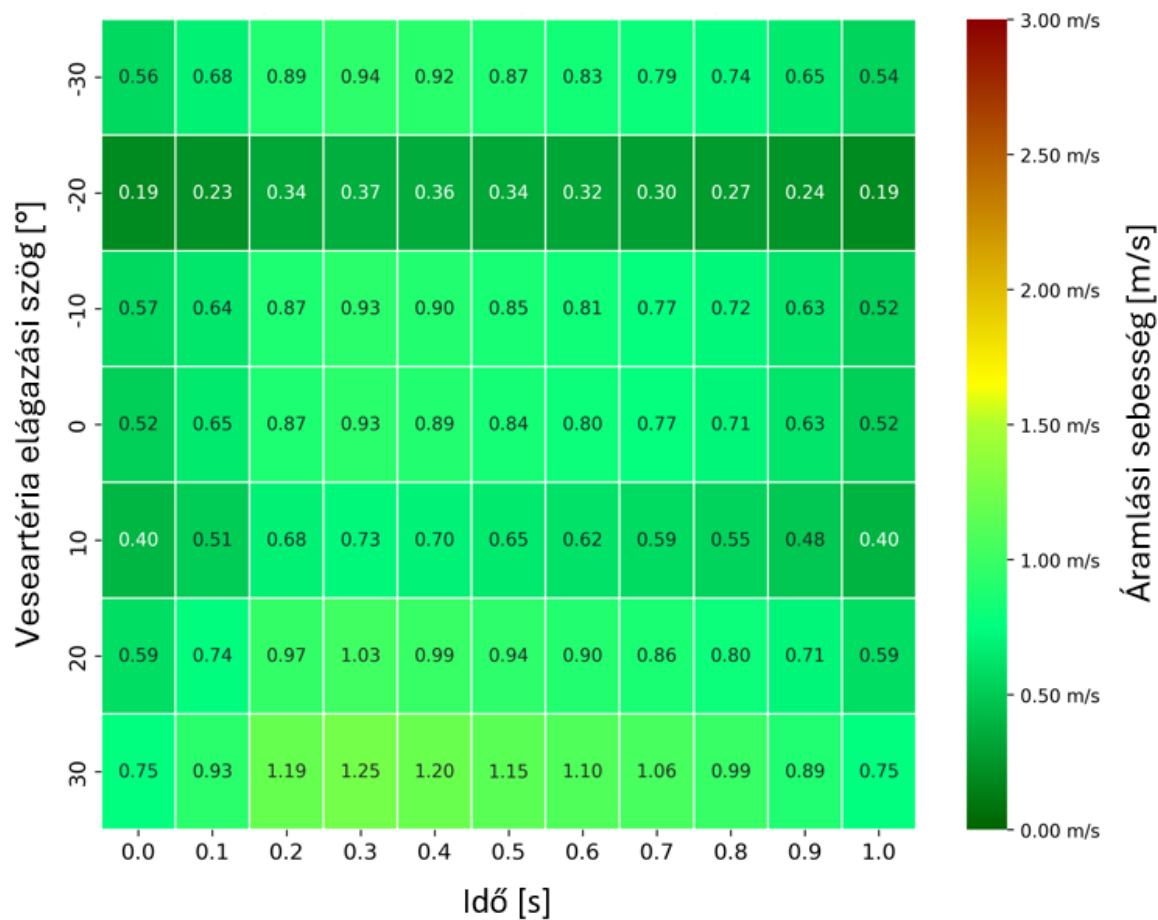
A.2. Modell 1 áramlási sebesség értékei

Az 1-es donor térfogat súlyozott maximum, átlag áramlási sebesség– szegmens 2



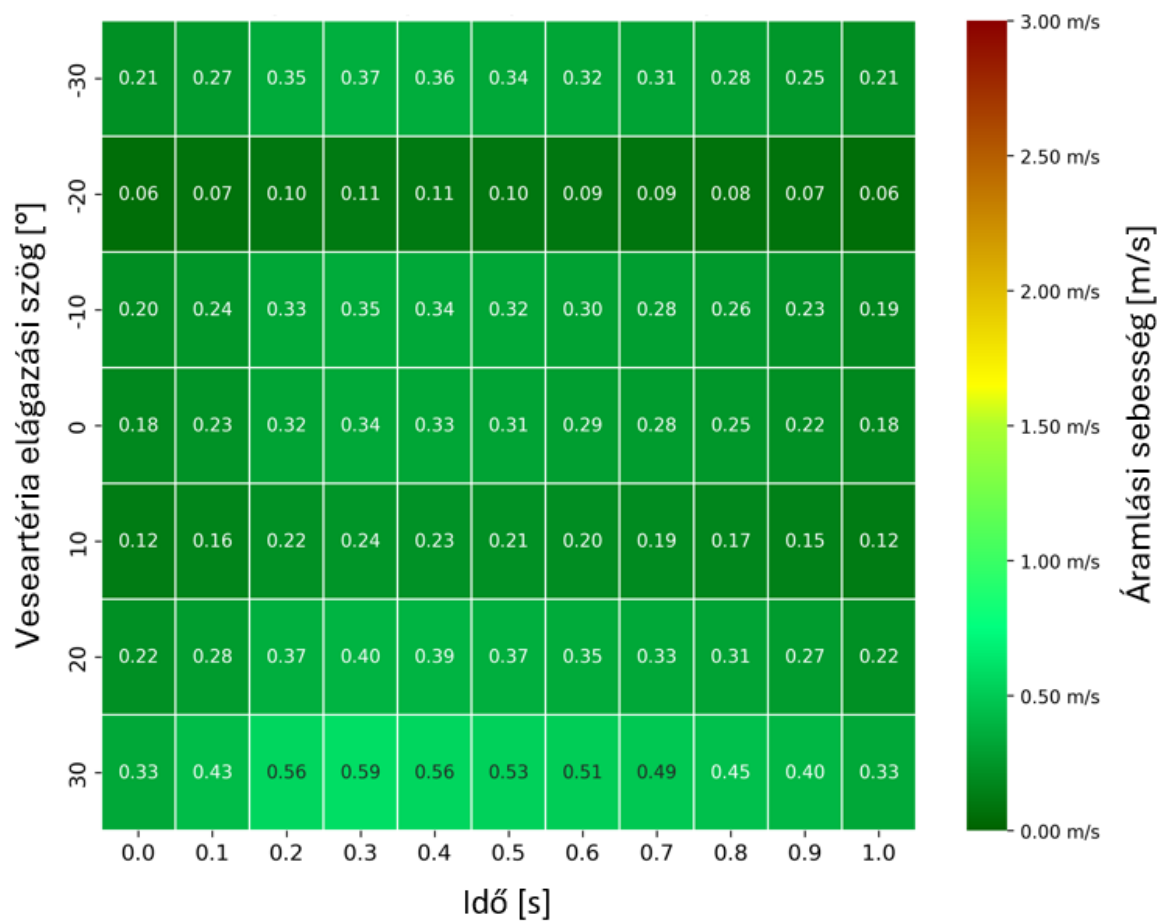
A.12. ábra. Térfogat súlyozott maximális és átlagos áramlási sebesség a 2. szegmensben az 1-es donor modell esetében. A zöld szín az egészséges tartományt jelöli 1,5 m/s, míg a piros a megemelkedett, krónikus szinteket mutatja.

Térfogat súlyozott maximális áramlási sebesség egy szívciklus alatt – szegmens 2



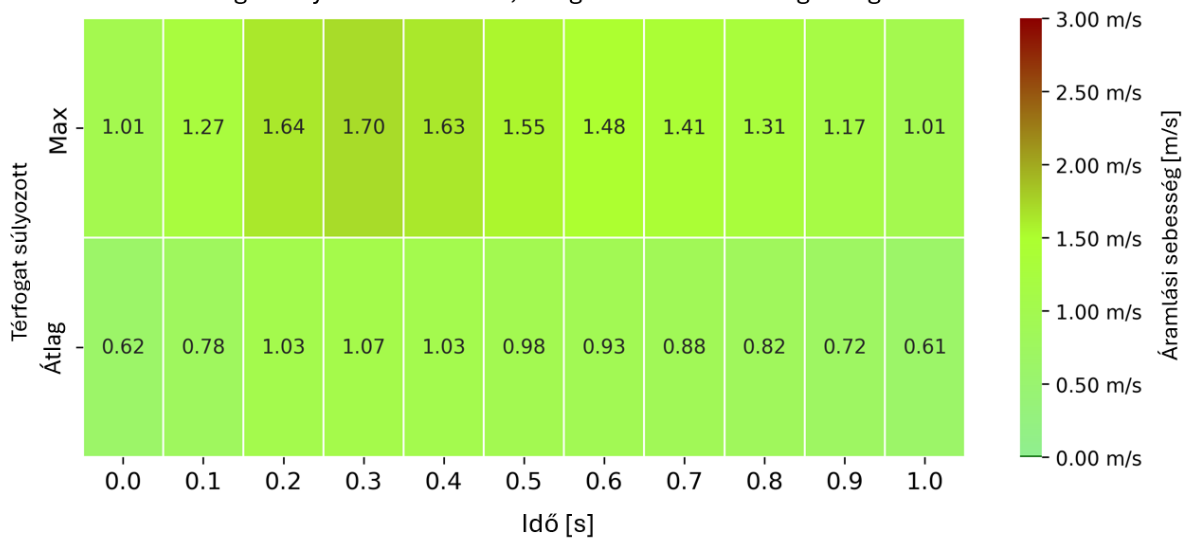
A.13. ábra. Térfogat súlyozott maximális áramlási sebesség a 2. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli 1,5 m/s, míg a piros a megemelkedett, krónikus szinteket mutatja.

Térfogat súlyozott átlagos áramlási sebesség egy szívciklus alatt – szegmens 2



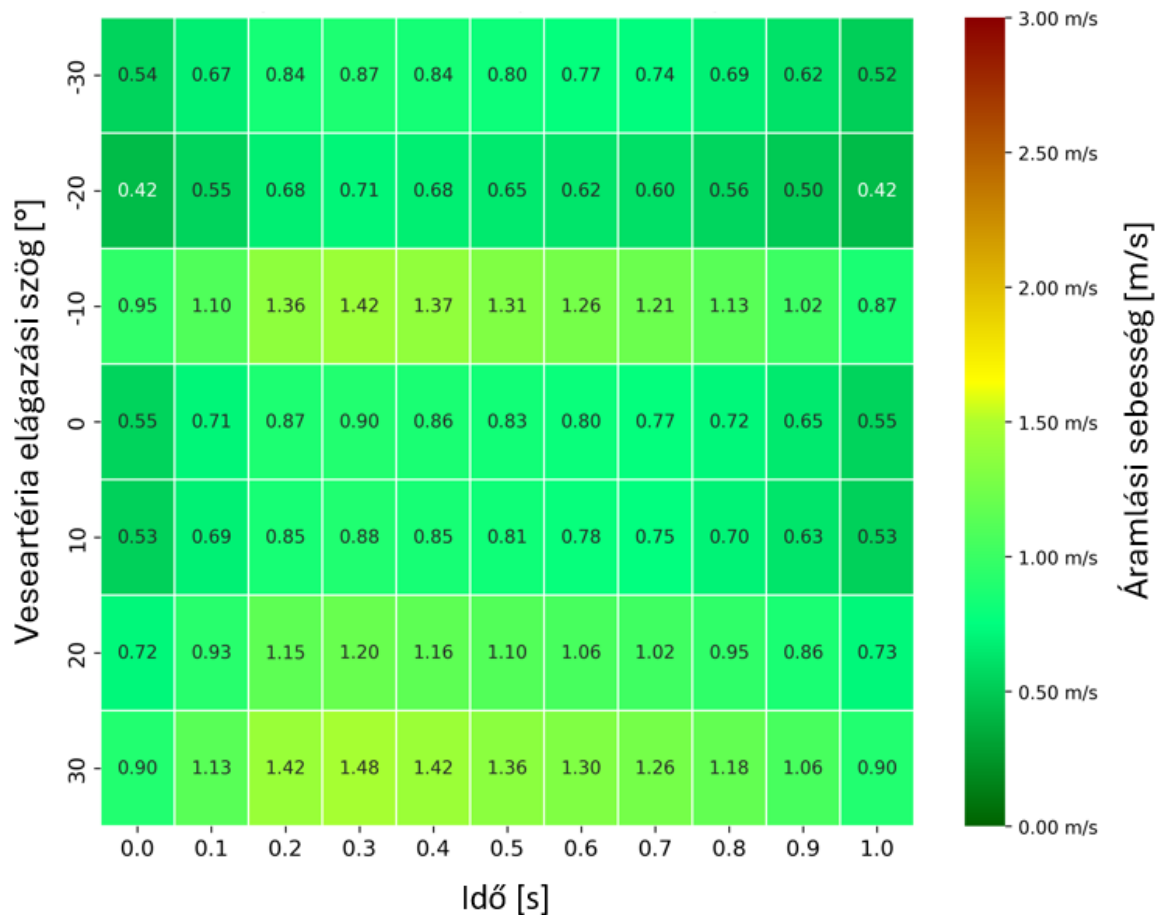
A.14. ábra. Térfogat súlyozott átlagos áramlási sebesség a 2. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli 1,5 m/s, míg a piros a megemelkedett, krónikus szinteket mutatja.

Az 1-es donor térfogat súlyozott maximum, átlag áramlási sebesség– szegmens 3



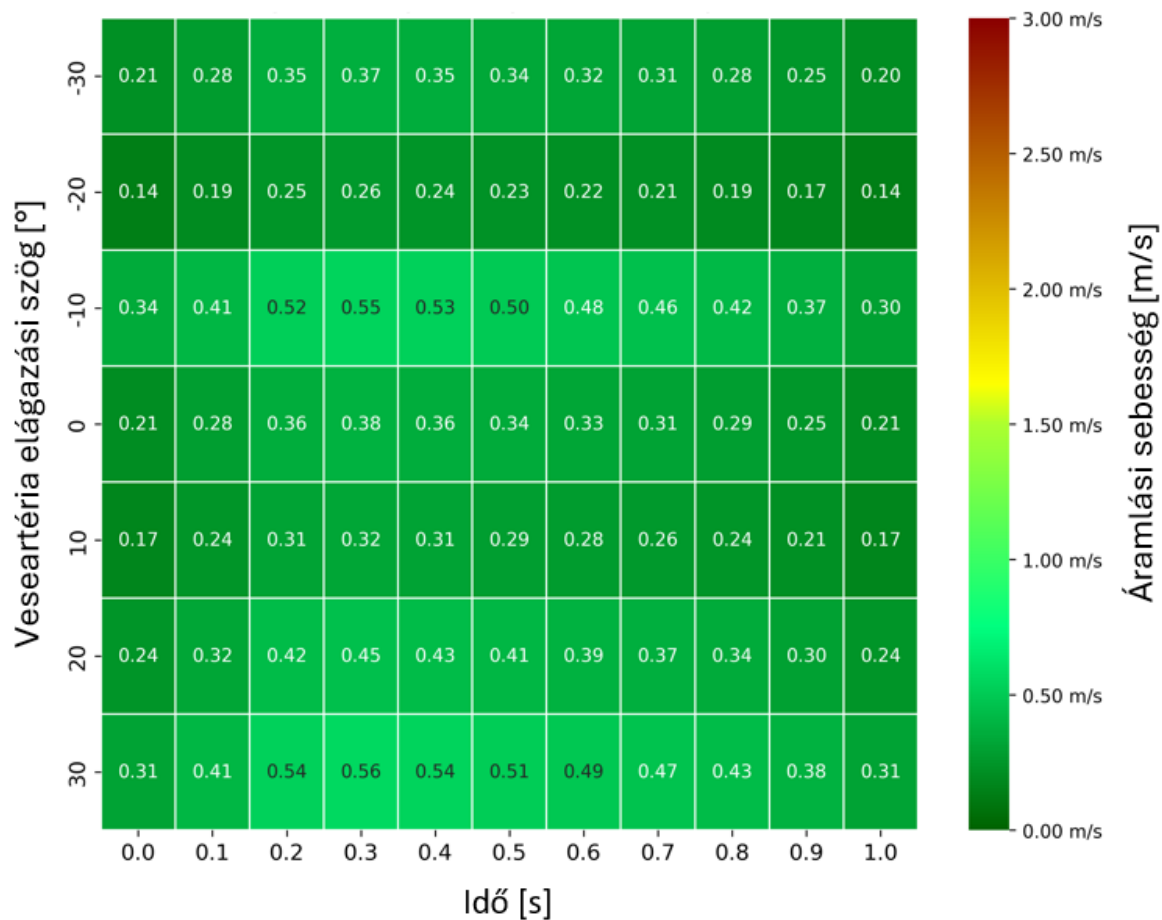
A.15. ábra. Térfogat súlyozott maximális és átlagos áramlási sebesség a 3. szegmensben az 1-es donor modell esetében. A zöld szín az egészséges tartományt jelöli 1,5 m/s, míg a piros a megemelkedett, krónikus szinteket mutatja.

Térfogat súlyozott maximális áramlási sebesség egy szív ciklus alatt – szegmens 3



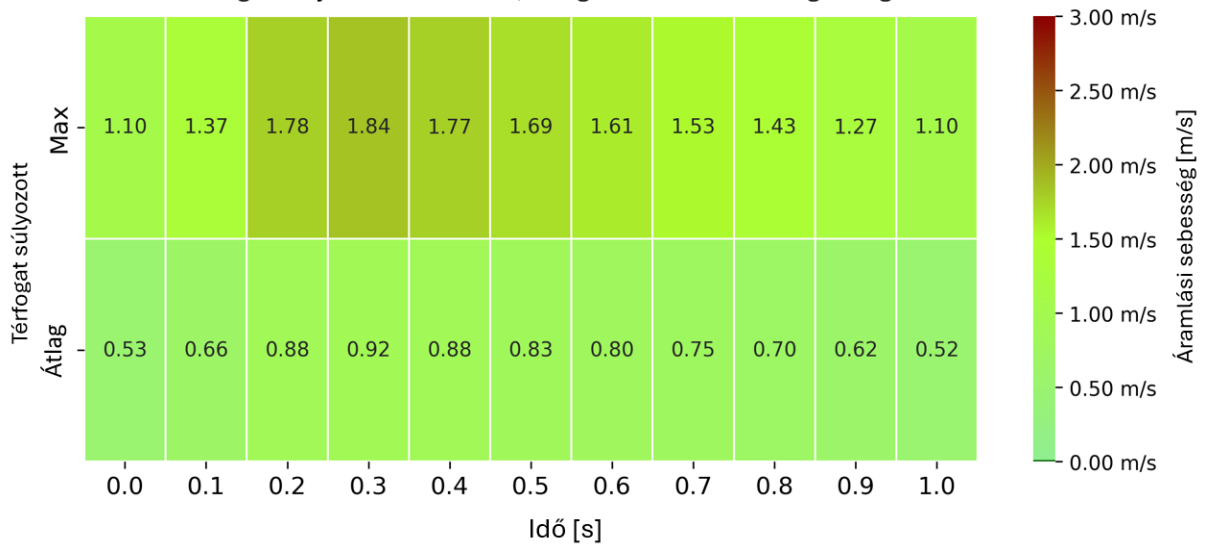
A.16. ábra. Térfogat súlyozott maximális áramlási sebesség a 3. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli 1,5 m/s, míg a piros a megemelkedett, krónikus szinteket mutatja.

Térfogat súlyozott átlagos áramlási sebesség egy szív ciklus alatt – szegmens 3



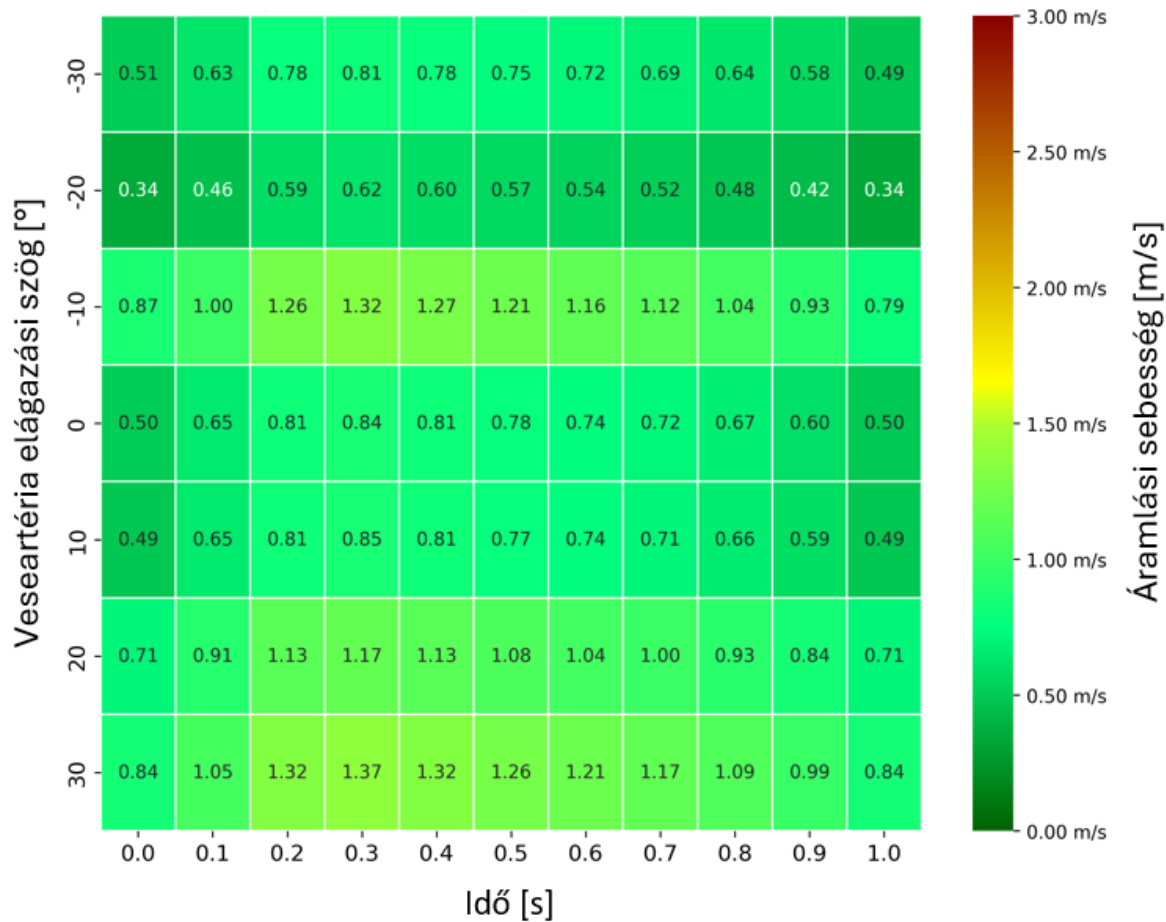
A.17. ábra. Térfogat súlyozott átlagos áramlási sebesség a 3. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli 1,5 m/s, míg a piros a megemelkedett, krónikus szinteket mutatja.

Az 1-es donor térfogat súlyozott maximum, átlag áramlási sebesség– szegmens 4



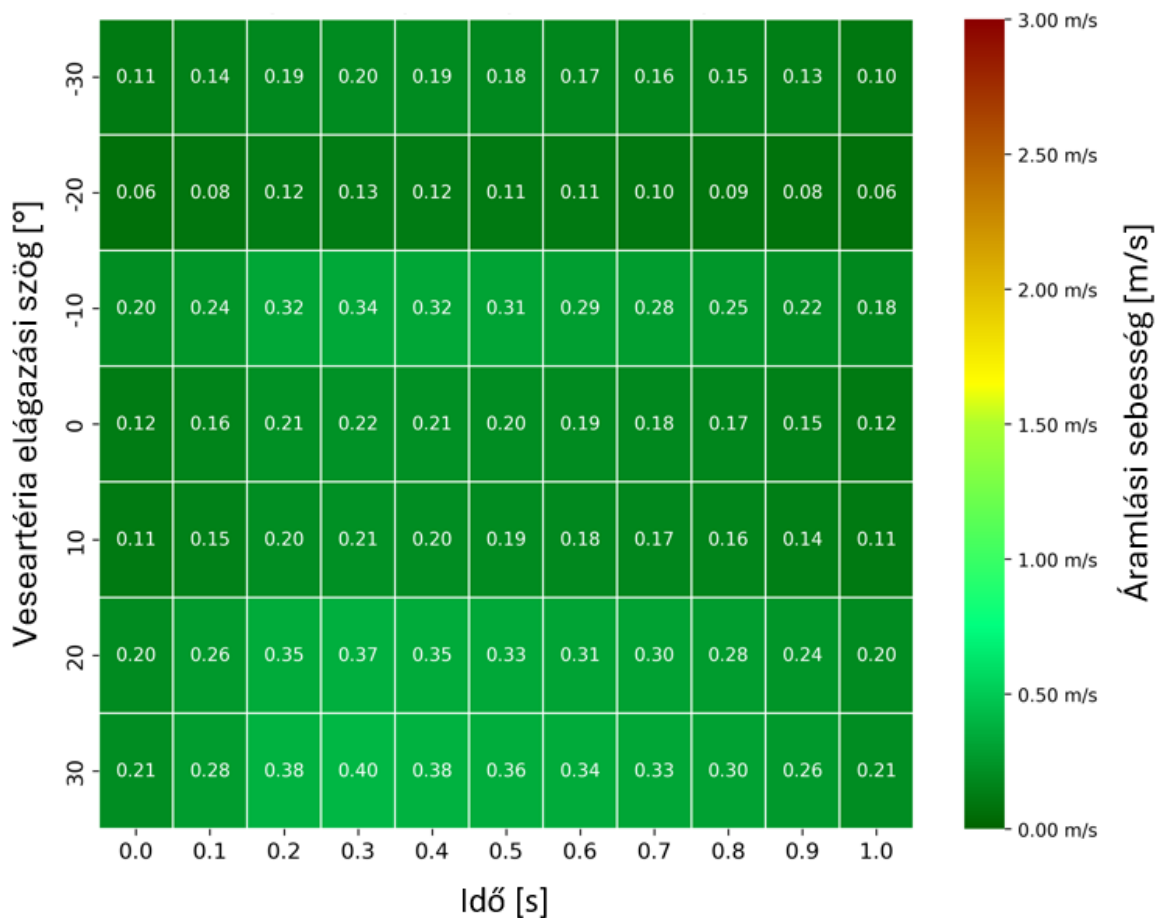
A.18. ábra. Térfogat súlyozott maximális és átlagos áramlási sebesség a 4. szegmensben az 1-es donor modell esetében. A zöld szín az egészséges tartományt jelöli 1,5 m/s, míg a piros a megemelkedett, krónikus szinteket mutatja.

Térfogat súlyozott maximális áramlási sebesség egy szívciklus alatt – szegmens 4



A.19. ábra. Térfogat súlyozott maximális áramlási sebesség a 4. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli 1,5 m/s, míg a piros a megemelkedett, krónikus szinteket mutatja.

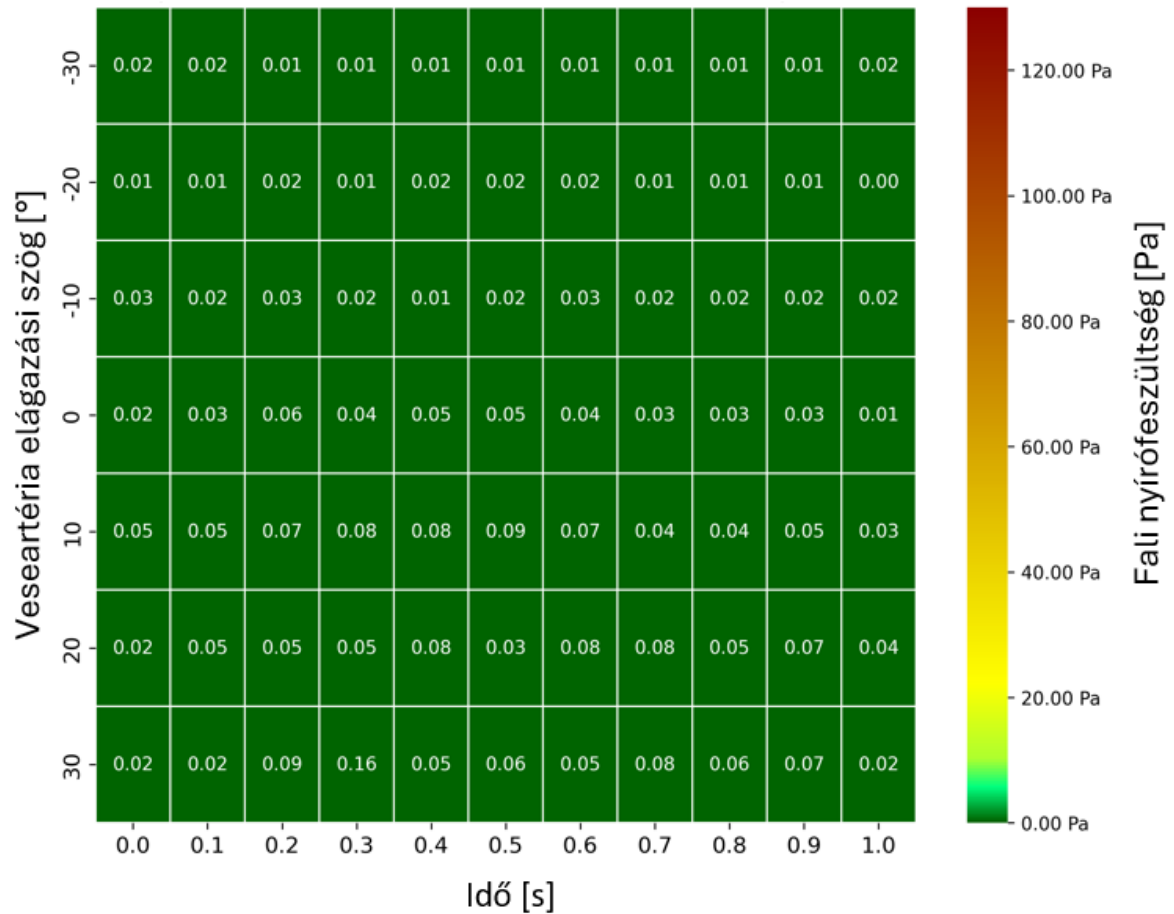
Térfogat súlyozott átlagos áramlási sebesség egy szív ciklus alatt – szegmens 4



A.20. ábra. Térfogat súlyozott átlagos áramlási sebesség a 4. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli 1,5 m/s, míg a piros a megemelkedett, krónikus szinteket mutatja.

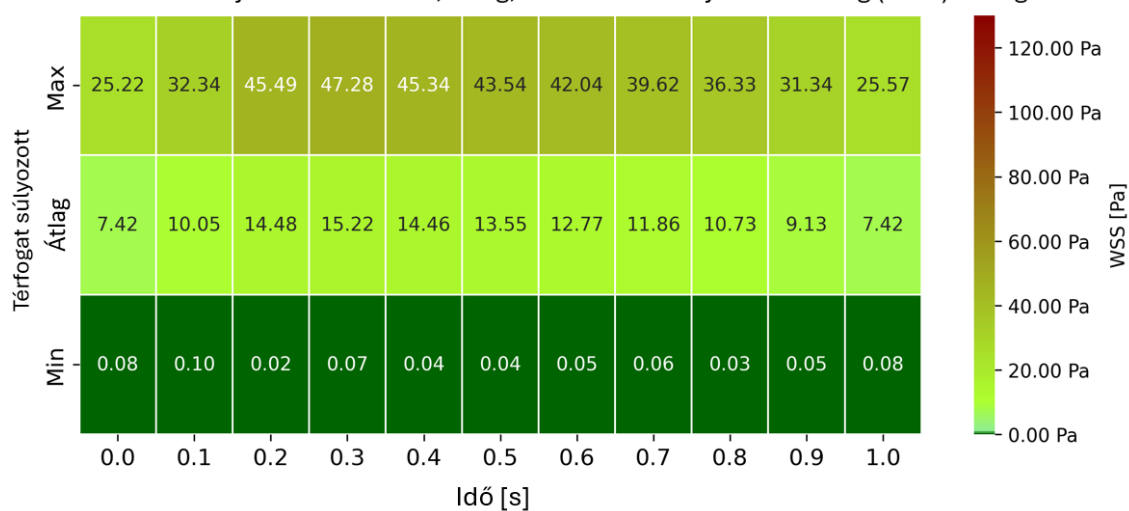
A.3. Modell 1 fali nyírófeszültség (WSS) értékei

Terület súlyozott minimális fali nyírófeszültség egy szív ciklus alatt – szegmens 1



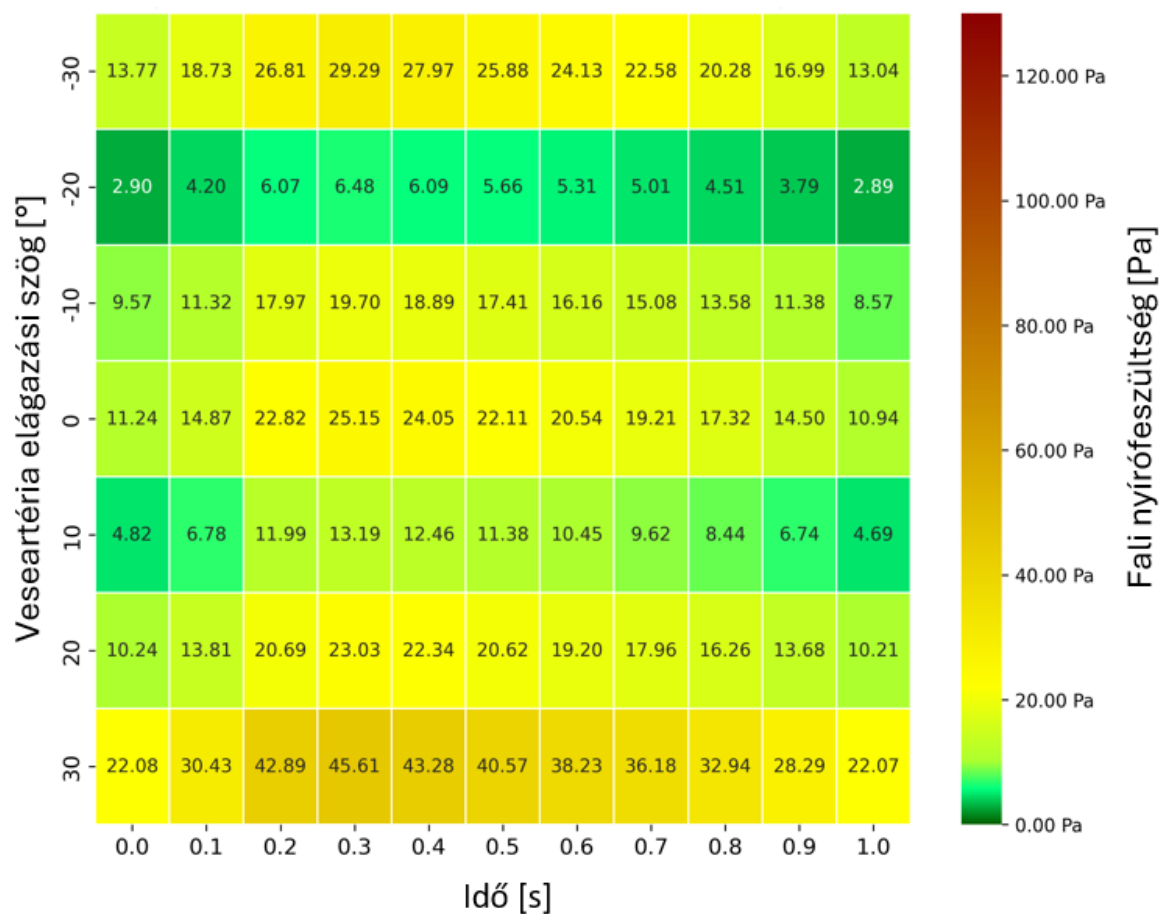
A.21. ábra. Terület súlyozott minimális fali nyírófeszültség (WSS) az 1. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli <7 Pa, míg a piros a megemelkedett, krónikus szinteket mutatja.

Az 1-es donor terület súlyozott maximum, átlag, minimum fali nyírófeszültség (WSS) – szegmens 2



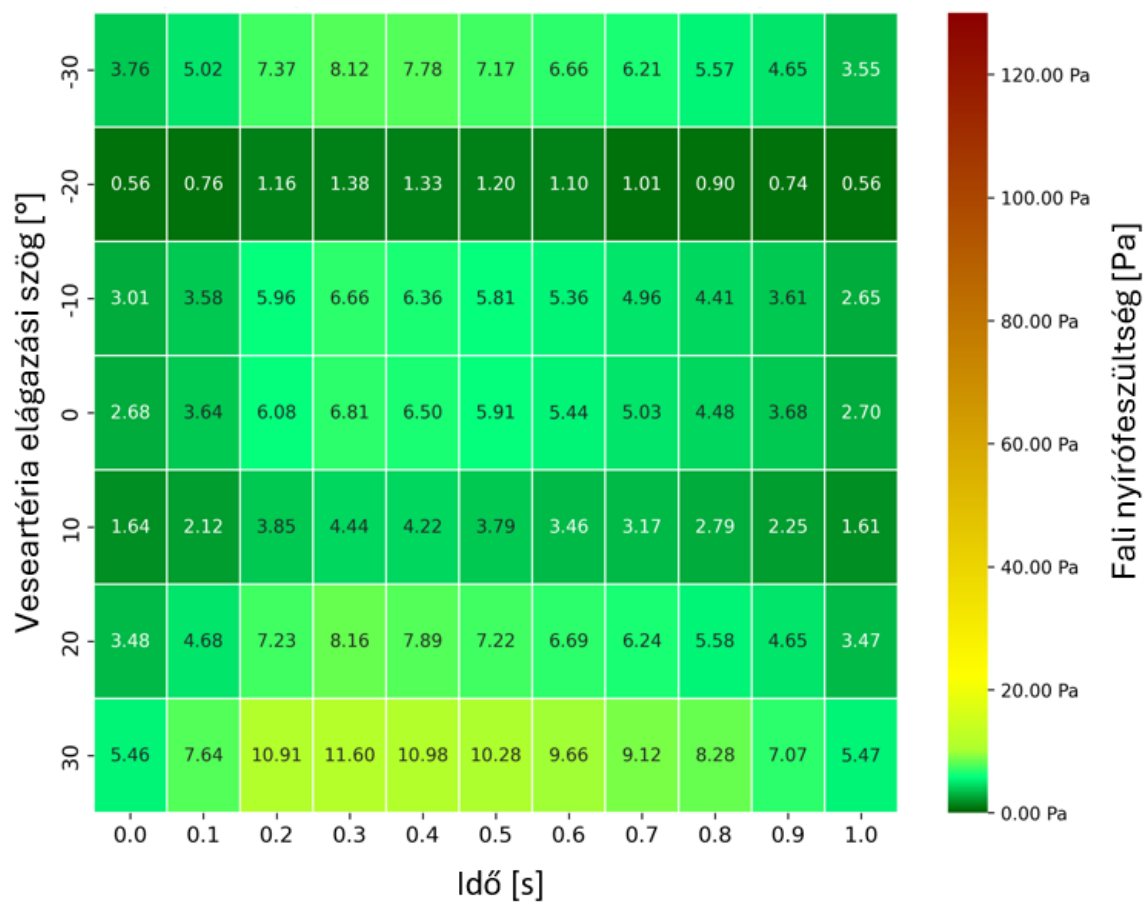
A.22. ábra. Terület súlyozott maximális, átlagos és minimális fali nyírófeszültség (WSS) a 2. szegmensben az 1-es donor modell esetében.

Terület súlyozott maximális fali nyírófeszültség egy szívciklus alatt – szegmens 2



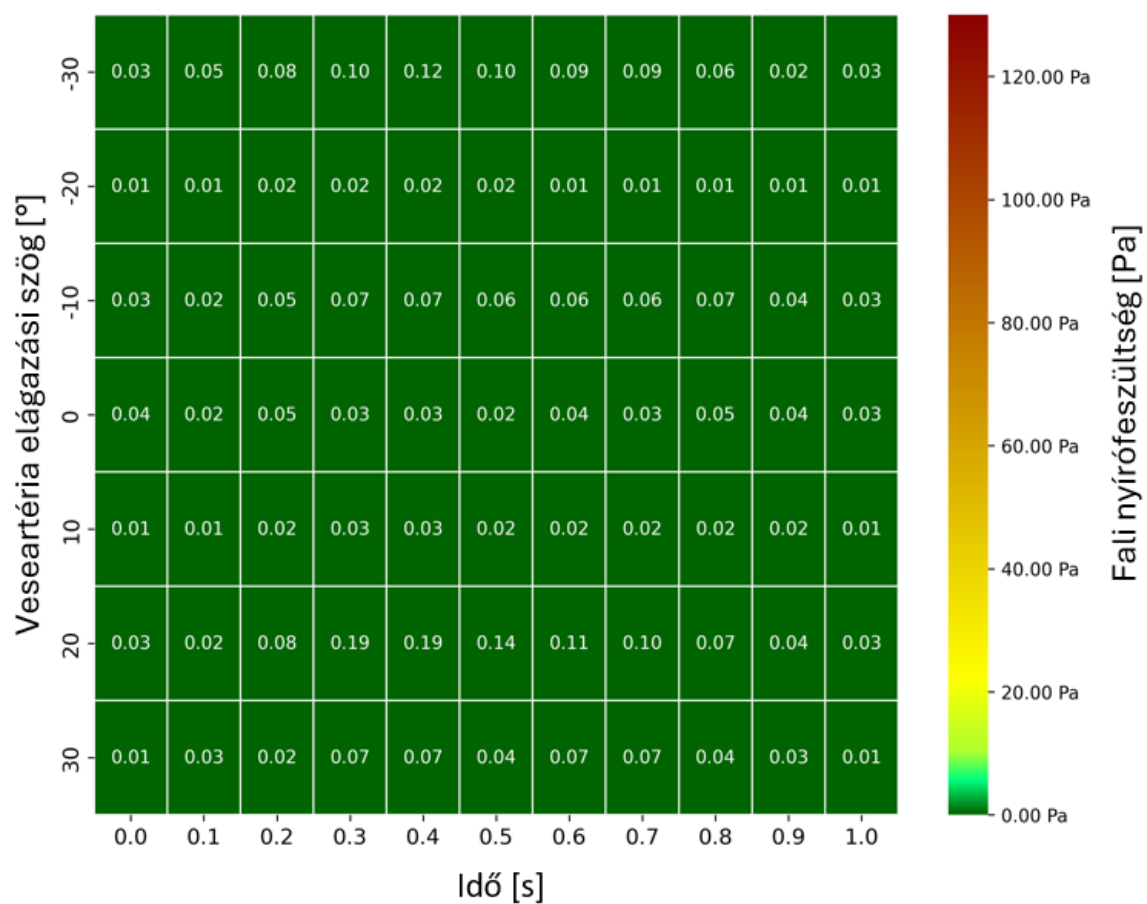
A.23. ábra. Terület súlyozott maximális fali nyírófeszültség (WSS) a 2. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli <7 Pa, míg a piros a megemelkedett, krónikus szinteket mutatja.

Terület súlyozott átlagos fali nyírófeszültség egy szív ciklus alatt – szegmens 2



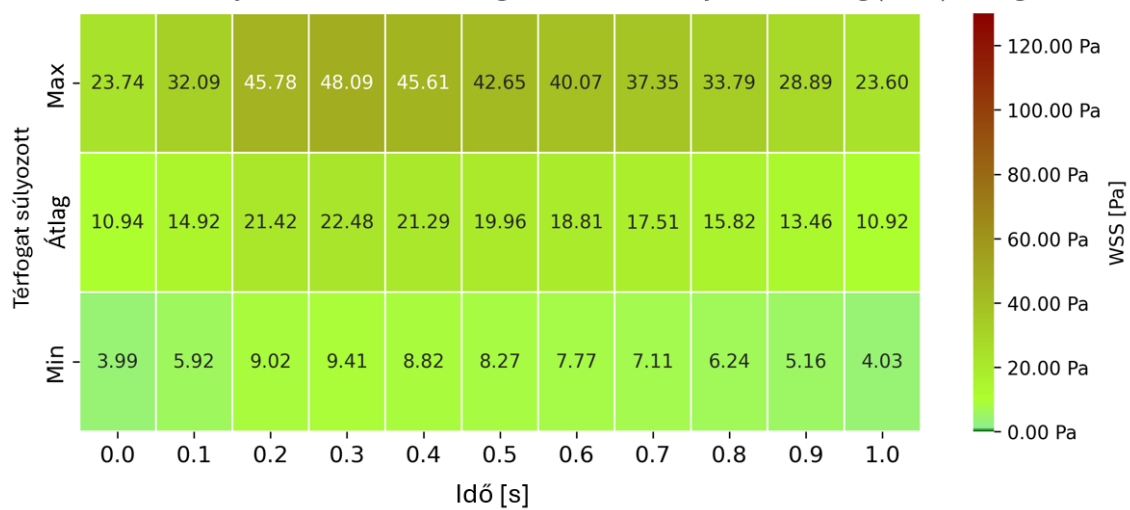
A.24. ábra. Terület súlyozott átlagos fali nyírófeszültség (WSS) a 2. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli <7 Pa, míg a piros a megemelkedett, krónikus szinteket mutatja.

Terület súlyozott minimális fali nyírófeszültség egy szívciklus alatt – szegmens 2



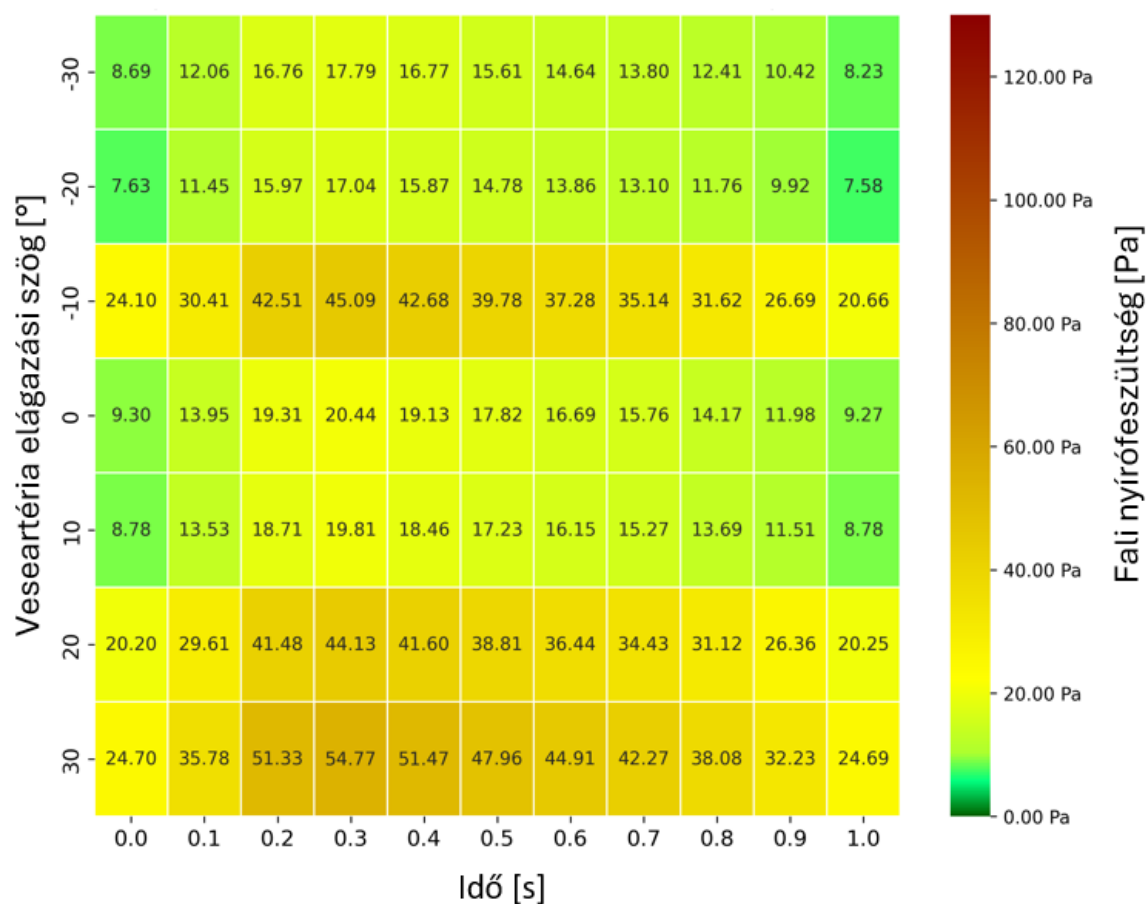
A.25. ábra. Terület súlyozott minimális fali nyírófeszültség (WSS) a 2. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli <7 Pa, míg a piros a megemelkedett, krónikus szinteket mutatja.

Az 1-es donor terület súlyozott maximum, átlag, minimum fali nyírófeszültség (WSS) – szegmens 3



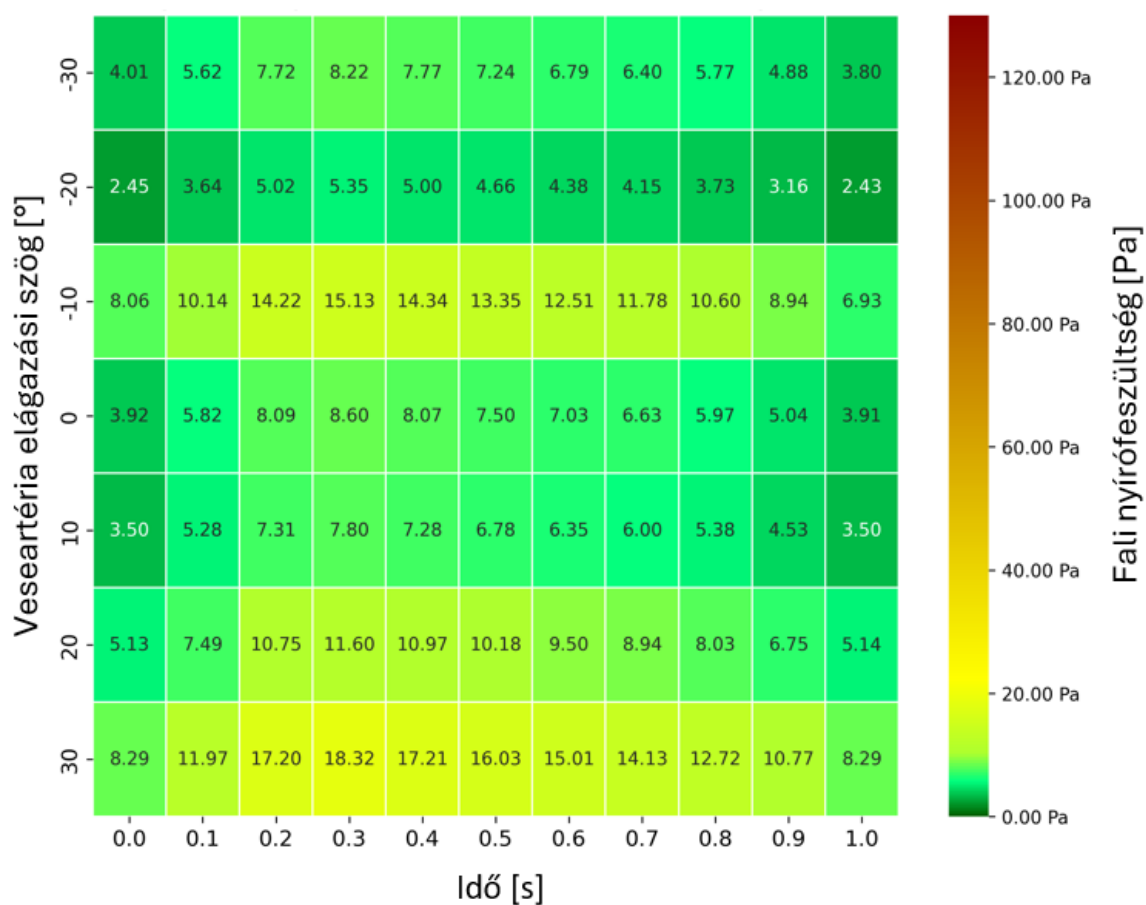
A.26. ábra. Terület súlyozott maximális, átlagos és minimális fali nyírófeszültség (WSS) a 3. szegmensben az 1-es donor modell esetében.

Terület súlyozott maximális fali nyírófeszültség egy szívciklus alatt – szegmens 3



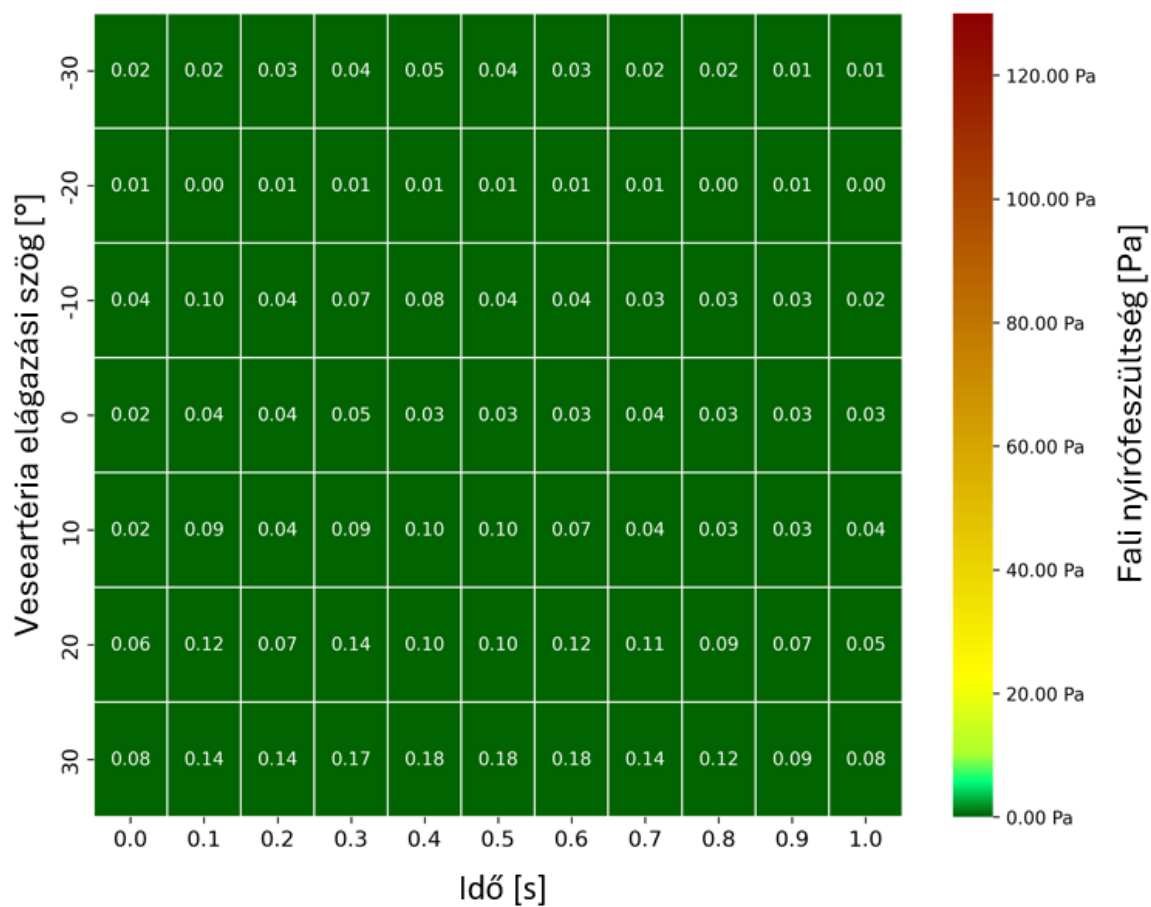
A.27. ábra. Terület súlyozott maximális fali nyírófeszültség (WSS) a 3. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli <7 Pa, míg a piros a megemelkedett, krónikus szinteket mutatja.

Terület súlyozott átlagos fali nyírófeszültség egy szívciklus alatt – szegmens 3



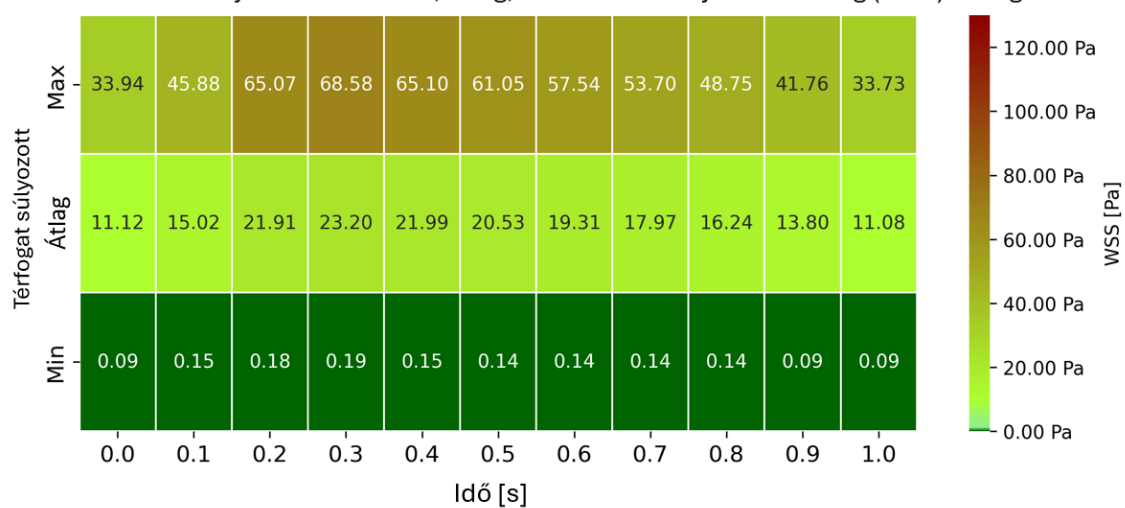
A.28. ábra. Terület súlyozott átlagos fali nyírófeszültség (WSS) a 3. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli <7 Pa, míg a piros a megemelkedett, krónikus szinteket mutatja.

Terület súlyozott minimális fali nyírófeszültség egy szív ciklus alatt – szegmens 3



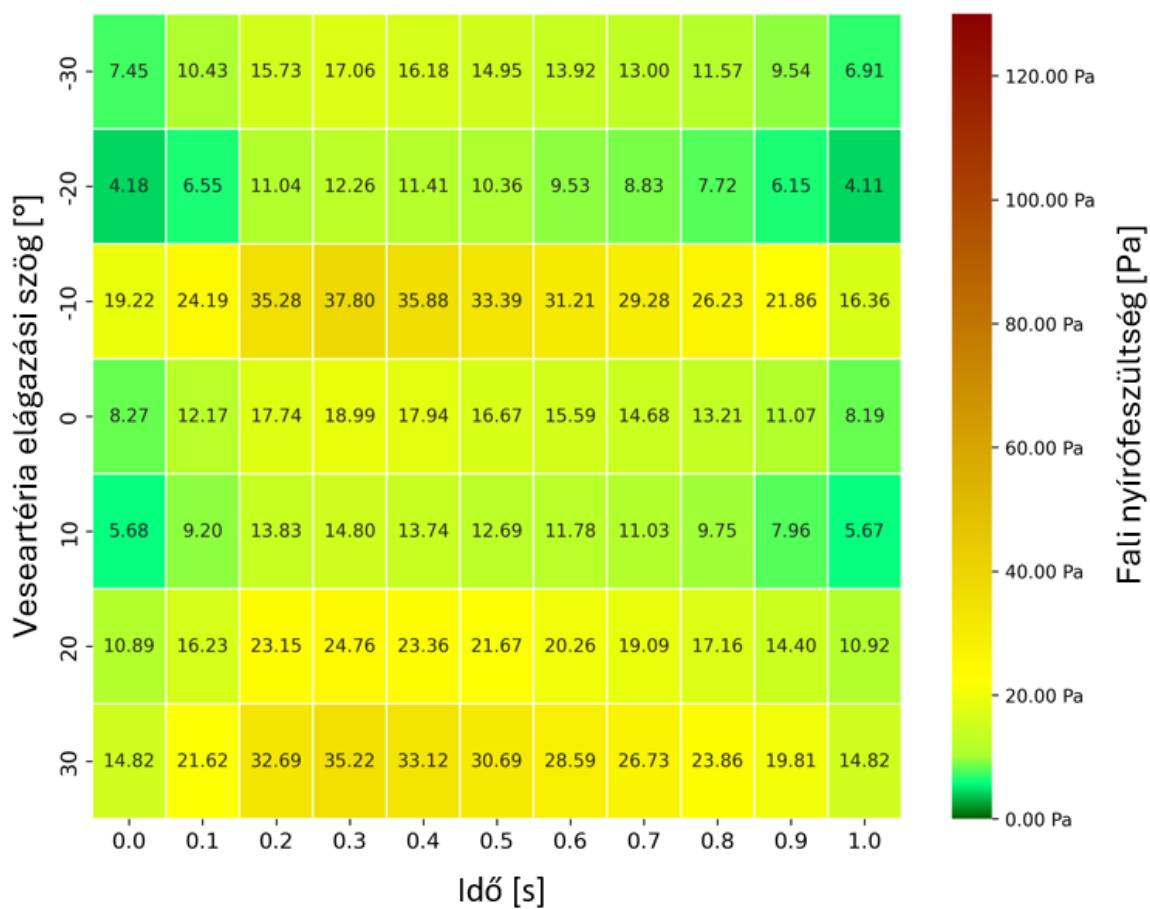
A.29. ábra. Terület súlyozott minimális fali nyírófeszültség (WSS) a 3. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli <7 Pa, míg a piros a megemelkedett, krónikus szinteket mutatja.

Az 1-es donor terület súlyozott maximum, átlag, minimum fali nyírófeszültség (WSS) – szegmens 4



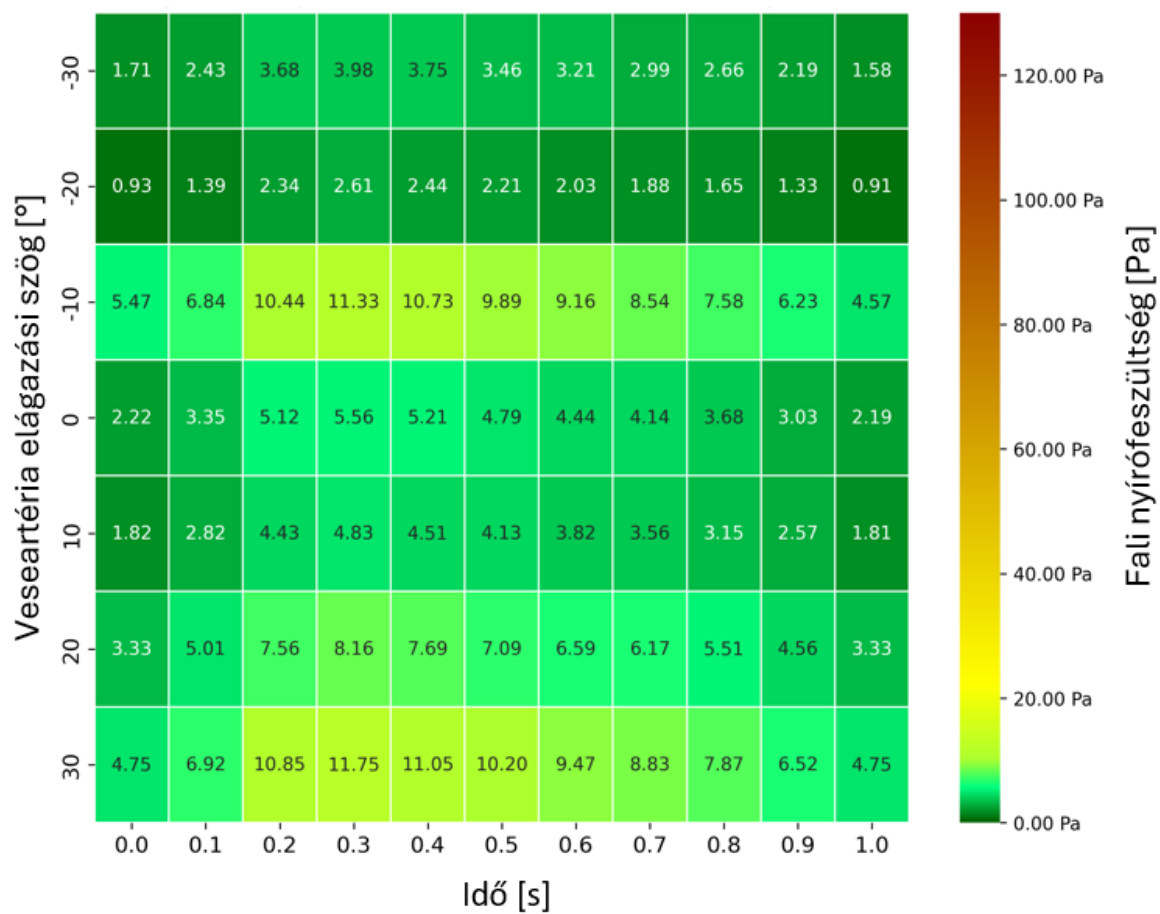
A.30. ábra. Terület súlyozott maximális, átlagos és minimális fali nyírófeszültség (WSS) a 4. szegmensben az 1-es donor modell esetében.

Terület súlyozott maximális fali nyírófeszültség egy szív ciklus alatt – szegmens 4



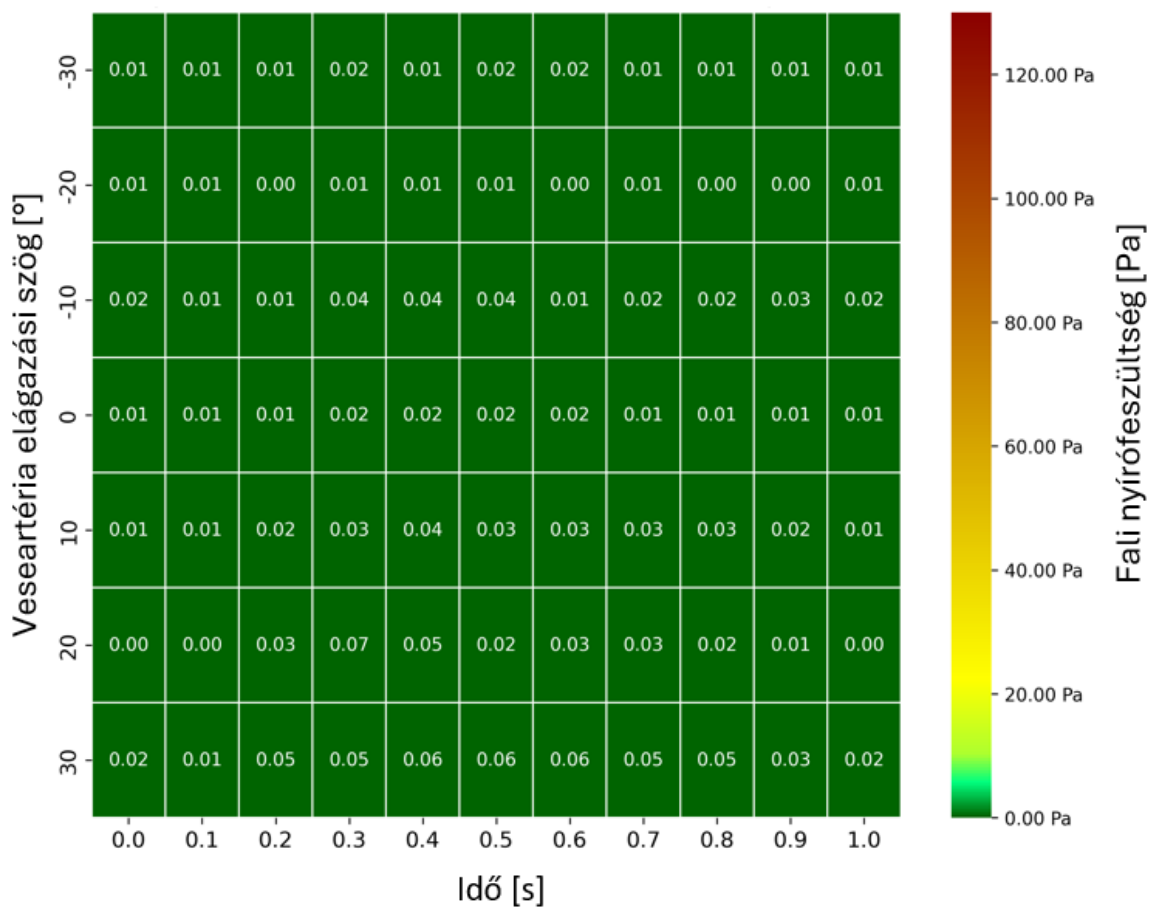
A.31. ábra. Terület súlyozott maximális fali nyírófeszültség (WSS) a 4. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli <7 Pa, míg a piros a megemelkedett, krónikus szinteket mutatja.

Terület súlyozott átlagos fali nyírófeszültség egy szív ciklus alatt – szegmens 4



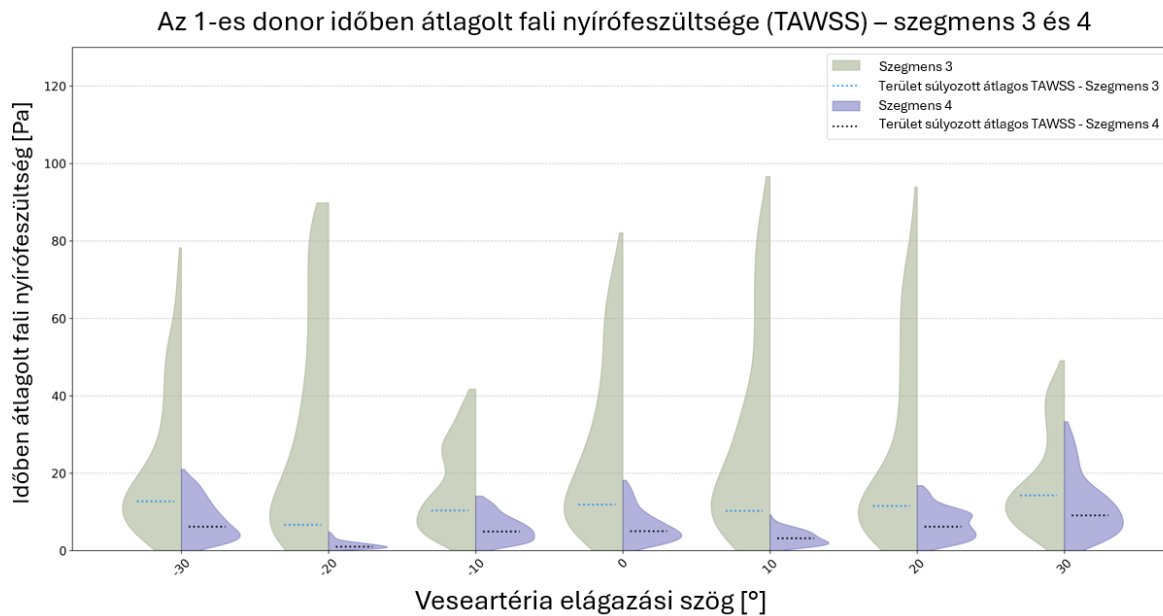
A.32. ábra. Terület súlyozott átlagos fali nyírófeszültség (WSS) a 4. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli <7 Pa, míg a piros a megemelkedett, krónikus szinteket mutatja.

Terület súlyozott minimális fali nyírófeszültség egy szívciklus alatt – szegmens 4



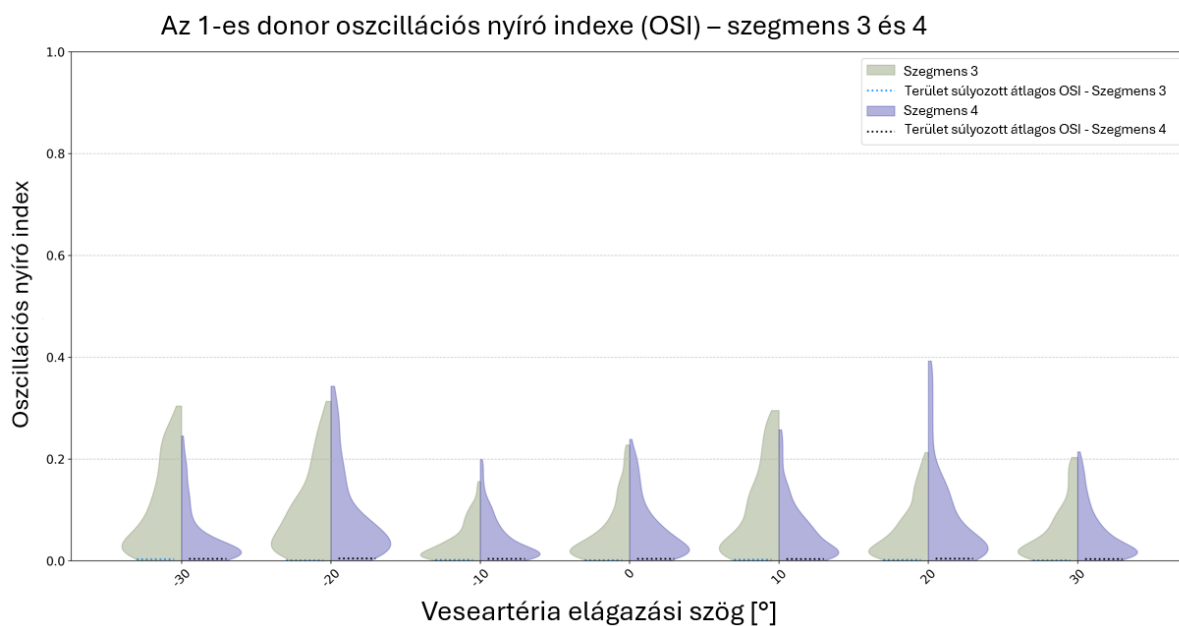
A.33. ábra. Terület súlyozott minimális fali nyírófeszültség (WSS) a 4. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján. A zöld szín az egészséges tartományt jelöli <7 Pa, míg a piros a megemelkedett, krónikus szinteket mutatja.

A.4. Modell 1 időben átlagolt fali nyírófeszültség (TAWSS) értékei



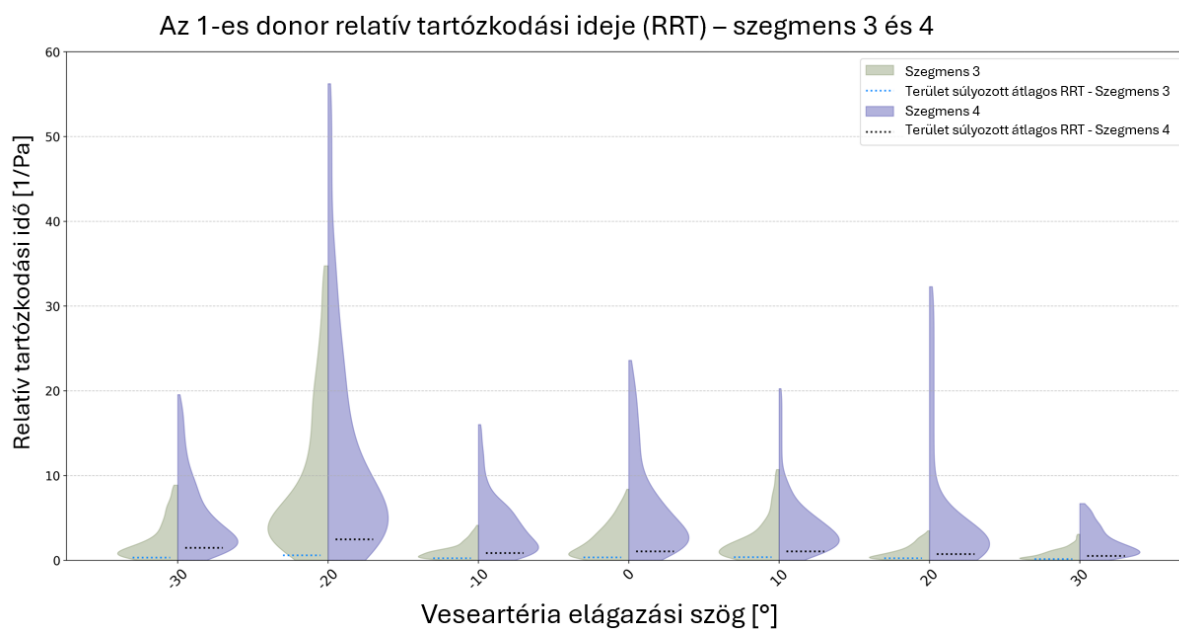
A.34. ábra. Terület súlyozott időben átlagolt fali nyírófeszültség (TAWSS) a 3. és 4. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján.

A.5. Modell 1 oszcillációs nyíró index értékei



A.35. ábra. Terület súlyozott oszcillációs nyíró index (OSI) a 3. és 4. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján.

A.6. Modell 1 relatív tartózkodási idő (RRT) értékei



A.36. ábra. Terület súlyozott relatív tartózkodási idő (RRT) a 3. és 4. szegmensben, különböző veseartéria elágazási szögek esetén, az 1-es donor modell alapján.

B. függelék

B.1. Az automatizált adatkinyerési folyamat forráskódja

```
1 #Import necessary libraries
2 import mph
3 import os
4 import re
5 import numpy as np
6
7 class SimulationProcessor:
8     def __init__(self, folder_path, client, domain, boundary, export_path,
9         part):
10         self.folder_path = folder_path
11         self.client = client
12         self.domain = domain
13         self.boundary = boundary
14         self.solnumindices = [f"{i}." for i in range(1, 256, 5)]
15         self.export_path = export_path
16         self.part = part
17
18     def process_simulations(self):
19         for item in os.listdir(self.folder_path):
20             item_path = os.path.join(self.folder_path, item)
21
22             # Check if the file has a .mph extension
23             if os.path.splitext(item)[1].lower() != '.mph':
24                 continue
25
26             # Load the simulation model using the COMSOL client
27             model = self.client.load(item_path)
```

```

27         final_name = self.extract_identififier(item)
28         if not final_name:
29             continue
30         java_model = model.java
31
32         # Process the different calculations
33         self.run_velocity_calculations(java_model, final_name)
34         self.run_pressure_calculations(java_model, final_name)
35         self.run_wss_calculations(java_model, final_name)
36         self.run_osi_calculations(java_model, final_name)
37         self.run_rrt_calculations(java_model, final_name)
38         self.run_tawss_calculations(java_model, final_name)
39
40     def extract_identififier(self, filename):
41         matches = re.findall(r'renalis_(-?\d+|full)_', filename)
42         if matches:
43             degree_part = matches[0]
44             return f'renalis_{degree_part}_'
45         return None
46
47     def create_table(self, java_model, table_name, comments):
48         table = java_model.result().table().create(table_name, 'Table')
49         table.comments(comments)
50         return table
51
52     def run_numerical_calculation(self, java_model, calc_type, selection,
53                                  expr, solnumindices, table_name, append=False):
54         calc = java_model.result().numerical().create(calc_type, selection)
55         calc.selection().set(self.domain if selection in ['AvVolume', '
56             MaxVolume', 'MinVolume'] else self.boundary)
57         calc.setIndex('expr', expr, 0)
58         calc.set('solnumindices', solnumindices)
59         calc.set('table', table_name)
60         if append:
61             calc.appendResult()
62         else:
63             calc.setResult()
64         calc.run()

```

```

64 def export_table(self, java_model, table_name, filename):
65     export = java_model.result().export().create(f'export_{table_name}',
66         'Table')
67     export.set('table', table_name)
68     export.set('source', 'table')
69     export.set('filename', filename)
70     export.run()
71
72 #Perform volume-weighted velocity calculations and export result
73 def run_velocity_calculations(self, java_model, final_name):
74     table = self.create_table(java_model, 'tbl1', 'Columns: 1 - Average,
75         2 - Maximum, 3 - Minimum')
76     self.run_numerical_calculation(java_model, 'av1', 'AvVolume', 'spf.U',
77         self.solnumindices, 'tbl1')
78     self.run_numerical_calculation(java_model, 'max1', 'MaxVolume', 'spf.
79         U', self.solnumindices, 'tbl1', append=True)
80     self.run_numerical_calculation(java_model, 'min1', 'MinVolume', 'spf.
81         U', self.solnumindices, 'tbl1', append=True)
82     self.export_table(java_model, 'tbl1', os.path.join(self.export_path,
83         f'{final_name}_velocity_{self.part}.csv'))
84
85 #Perform volume-weighted velocity calculations and export result
86 def run_pressure_calculations(self, java_model, final_name):
87     table = self.create_table(java_model, 'tbl2', 'Columns: 1 - Average,
88         2 - Maximum, 3 - Minimum')
89     self.run_numerical_calculation(java_model, 'av2', 'AvVolume', 'p',
90         self.solnumindices, 'tbl2')
91     self.run_numerical_calculation(java_model, 'max2', 'MaxVolume', 'p',
92         self.solnumindices, 'tbl2', append=True)
93     self.run_numerical_calculation(java_model, 'min2', 'MinVolume', 'p',
94         self.solnumindices, 'tbl2', append=True)
95     self.export_table(java_model, 'tbl2', os.path.join(self.export_path,
96         f'{final_name}_pressure_{self.part}.csv'))
97
98 #Perform area-weighted wall shear stress calculations and export result
99 def run_wss_calculations(self, java_model, final_name):
100     expr = 'sqrt(spf.K_stressx^2 + spf.K_stressy^2 + spf.K_stressz^2)'
101     table = self.create_table(java_model, 'tbl3', 'Columns: 1 - Average,
102         2 - Maximum, 3 - Minimum')

```

```

91     self.run_numerical_calculation(java_model, 'av3', 'AvSurface', expr,
92                                   self.solnumindices, 'tbl3')
93     self.run_numerical_calculation(java_model, 'max3', 'MaxSurface', expr
94                                   , self.solnumindices, 'tbl3', append=True)
95     self.run_numerical_calculation(java_model, 'min3', 'MinSurface', expr
96                                   , self.solnumindices, 'tbl3', append=True)
97     self.export_table(java_model, 'tbl3', os.path.join(self.export_path,
98                                   f'{final_name}_wss_{self.part}.csv'))
99
100 #Perform area-weighted oscillatory shear index calculations and export
101 #result
102 def run_osi_calculations(self, java_model, final_name):
103     expr = '0.5 * (1 - (sqrt(timeint(2,3,spf.K_stressx)^2 + timeint(2,3,
104                                   spf.K_stressy)^2 + timeint(2,3,spf.K_stressz)^2) / timeint(2,3,
105                                   sqrt(spf.K_stressx^2 + spf.K_stressy^2 + spf.K_stressz^2))))'
106     table = self.create_table(java_model, 'tbl4', 'Columns: 1 - Average,
107                                   2 - Maximum, 3 - Minimum')
108     self.run_numerical_calculation(java_model, 'av4', 'AvSurface', expr,
109                                   [76.], 'tbl4')
110     self.run_numerical_calculation(java_model, 'max4', 'MaxSurface', expr
111                                   , [76.], 'tbl4', append=True)
112     self.run_numerical_calculation(java_model, 'min4', 'MinSurface', expr
113                                   , [76.], 'tbl4', append=True)
114     self.export_table(java_model, 'tbl4', os.path.join(self.export_path,
115                                   f'{final_name}_osi_{self.part}.csv'))
116
117 #Perform area-weighted relative residence time calculations and export
118 #result
119 def run_rrt_calculations(self, java_model, final_name):
120     expr = '1/((1 - 2*(0.5 * (1 - (sqrt(timeint(2,3,spf.K_stressx)^2 +
121                                   timeint(2,3,spf.K_stressy)^2 + timeint(2,3,spf.K_stressz)^2) /
122                                   timeint(2,3,sqrt(spf.K_stressx^2 + spf.K_stressy^2 + spf.
123                                   K_stressz^2)))))))*((1/1)*(timeavg(2,3,sqrt(spf.K_stressx^2 + spf.
124                                   K_stressy^2 + spf.K_stressz^2))))'
125     table = self.create_table(java_model, 'tbl5', 'Columns: 1 - Average,
126                                   2 - Maximum, 3 - Minimum')
127     self.run_numerical_calculation(java_model, 'av5', 'AvSurface', expr,
128                                   [76.], 'tbl5')

```

```

110     self.run_numerical_calculation(java_model, 'max5', 'MaxSurface', expr
    , [76.], 'tbl5', append=True)
111     self.run_numerical_calculation(java_model, 'min5', 'MinSurface', expr
    , [76.], 'tbl5', append=True)
112     self.export_table(java_model, 'tbl5', os.path.join(self.export_path,
    f'{final_name}_rrt_{self.part}.csv'))
113
114     #Perform area-weighted time-averaged wall shear stress calculations and
    export result
115     def run_tawss_calculations(self, java_model, final_name):
116         expr = '1/1 * timeavg(2,3,sqrt(spf.K_stressx^2 + spf.K_stressy^2 +
    spf.K_stressz^2))'
117         table = self.create_table(java_model, 'tbl6', 'Columns: 1 - Average,
    2 - Maximum, 3 - Minimum')
118         self.run_numerical_calculation(java_model, 'av6', 'AvSurface', expr,
    [76.], 'tbl6')
119         self.run_numerical_calculation(java_model, 'max6', 'MaxSurface', expr
    , [76.], 'tbl6', append=True)
120         self.run_numerical_calculation(java_model, 'min6', 'MinSurface', expr
    , [76.], 'tbl6', append=True)
121         self.export_table(java_model, 'tbl6', os.path.join(self.export_path,
    f'{final_name}_tawss_{self.part}.csv'))
122
123     # Start COMSOL client
124     client = mph.start()
125     folder_path = 'D:/Simulations'
126     domain = ['domain number'] #domain selection
127     boundary = ['boundary number'] #boundary selection
128     part = 4 #section selection
129     export_path = 'D:/Simulations/results'
130
131     processor = SimulationProcessor(folder_path, client, domain, boundary,
    export_path, part)
132     processor.process_simulations()

```