In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from matgen.base import CellComplex
```

In [ ]:
```
# Header: ^P::Cu-0.1Cr-0.1Zr ST+AT ECAP-C 20 C 1p-4 cleaned-NCIC::All data::
#
# Column 1: Integer identifying grain
# Column 2-4: Average orientation (phi1, PHI, phi2) in degrees
# Column 5-7: Average orientation (phi1, PHI, phi2) in radians
# Column 8-9: Average Position (x, y) in microns
# Column 10: Edge grain (1) or interior grain (0)
# Column 11: Diameter of grain in microns
```

In [6]:
```
data1 = pd.read_csv('/Users/v94623eb2/Documents/ExpDataAnalysis/Experimental
           skiprows=8, sep='\s+', names=['id', 'phi1_degree', 'PHI_degree',
```

In [7]:
```
data1.info()
data1.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 992 entries, 0 to 991
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   id           992 non-null    int64
 1   phi1_degree  992 non-null    float64
 2   PHI_degree   992 non-null    float64
 3   phi2_degree  992 non-null    float64
 4   phi1_rad     992 non-null    float64
 5   PHI_rad      992 non-null    float64
 6   phi2_rad     992 non-null    float64
 7   x            992 non-null    float64
 8   y            992 non-null    float64
 9   is_edge      992 non-null    int64
 10  D            992 non-null    float64
dtypes: float64(9), int64(2)
memory usage: 85.4 KB
```

Out[7]:

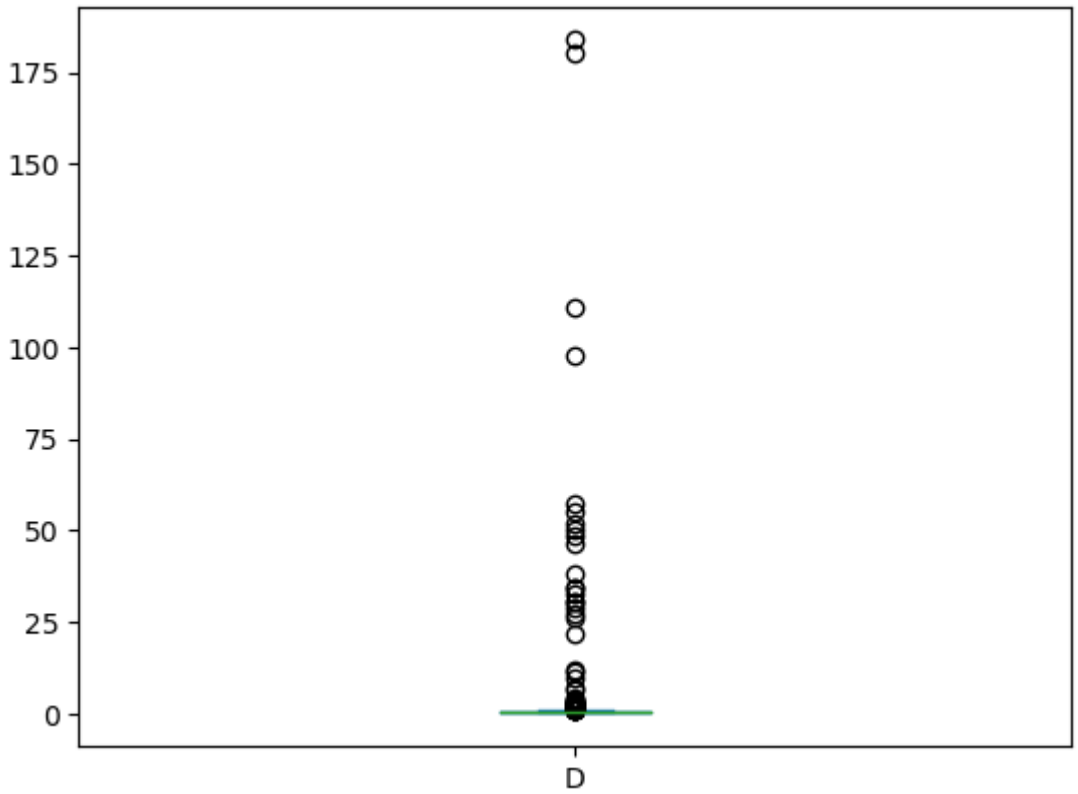
| | id | phi1_degree | PHI_degree | phi2_degree | phi1_rad | PHI_rad | phi2_rad | x | y |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 6.944 | 46.965 | 351.34 | 0.12119 | 0.81970 | 6.13196 | 28.461 | 14.357 |
| **1** | 2 | 52.267 | 23.604 | 295.28 | 0.91223 | 0.41196 | 5.15361 | 1.628 | 0.203 |
| **2** | 3 | 193.599 | 42.040 | 191.87 | 3.37893 | 0.73374 | 3.34870 | 12.905 | 0.396 |
| **3** | 5 | 248.408 | 51.530 | 124.97 | 4.33553 | 0.89936 | 2.18116 | 101.028 | 93.695 |
| **4** | 6 | 26.442 | 45.930 | 347.80 | 0.46149 | 0.80163 | 6.07028 | 86.400 | 0.000 |

In [8]:
```
data1.D.describe()
```

Out[8]:
```
count    992.000000
mean       1.792762
std       10.605383
min        0.300000
25%        0.300000
50%        0.470000
75%        0.660000
max      183.770000
Name: D, dtype: float64
```

In [13]:
```
data1.D.plot(kind='box')
```

Out[13]:
```
<AxesSubplot:>
```




In [19]:
```
data1.x.describe()
```

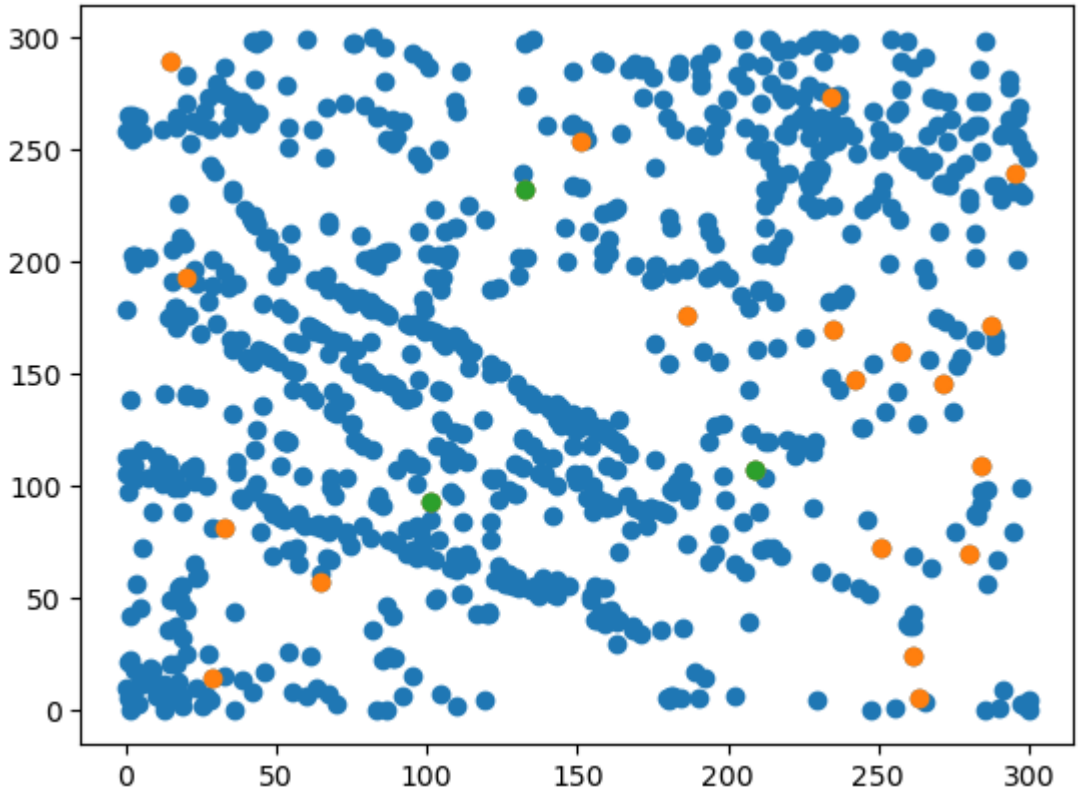
Out[19]:
```
count    992.000000
mean     133.523304
std       86.138826
min        0.033000
25%       57.448250
50%      124.553000
75%      208.936250
max      299.900000
Name: x, dtype: float64
```

In [20]:
```
data1.y.describe()
```

Out[20]:
```
count    992.000000
mean     154.933476
std       85.112094
min        0.000000
25%       88.785250
50%      154.232500
75%      232.063500
max      299.905000
Name: y, dtype: float64
```

In [49]:
```
plt.scatter(data1.x, data1.y)
plt.scatter(bigdata1.x, bigdata1.y)
plt.scatter(biggestdata1.x, biggestdata1.y)
```

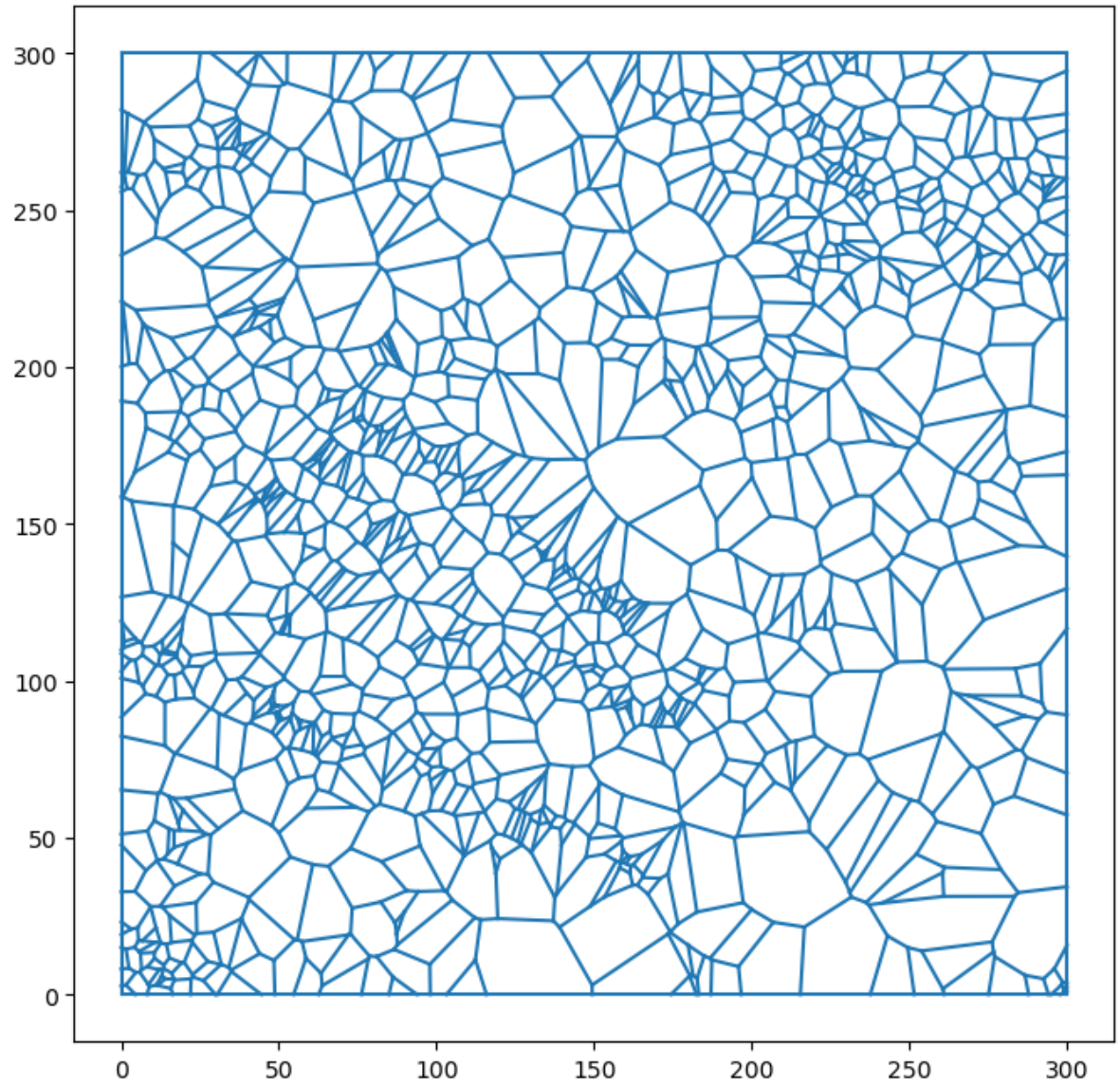
Out[49]:
```
<matplotlib.collections.PathCollection at 0x7fb69131faf0>
```




In [16]:
```
c1 = CellComplex.from_tess_file('/Users/v94623eb2/Documents/ExpDataAnalysis/
```

In [18]:
```
c1.plot_edges(color='C0')
```

Out[18]:
```
<AxesSubplot:>
```




In [8]:
```
import math
import numpy as np
```

In [9]:
```
areas_neper = np.loadtxt('/Users/v94623eb2/Documents/ExpDataAnalysis/Experim
```

In [29]:
```
eq_diams = [math.sqrt(4 * area / math.pi) for area in areas]
```

In [31]:
```
pd.DataFrame(eq_diams).describe()
```

Out[31]:

| | 0 |
|---|---|
| **count** | 992.000000 |
| **mean** | 9.467637 |
| **std** | 5.089759 |
| **min** | 0.599169 |
| **25%** | 5.649042 |
| **50%** | 8.673054 |
| **75%** | 12.077238 |
| **max** | 32.357945 |

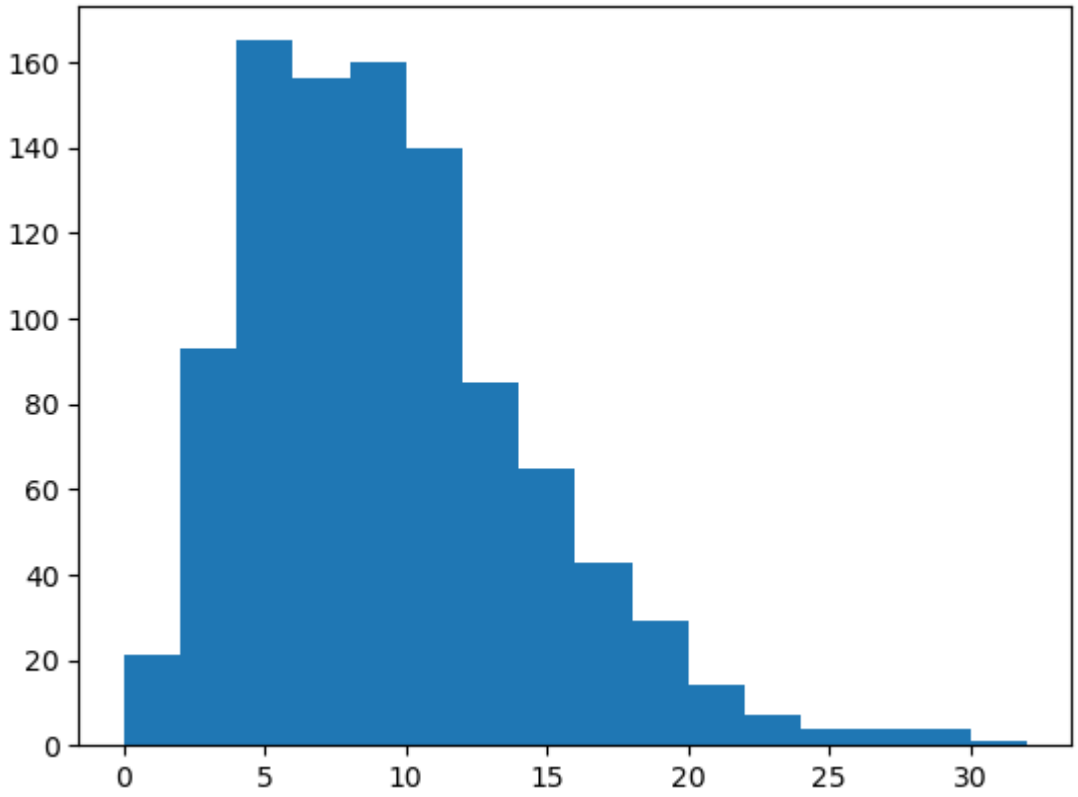
In [32]:
```
data1[['D']].describe()
```

Out[32]:

| | D |
|---|---|
| **count** | 992.000000 |
| **mean** | 1.792762 |
| **std** | 10.605383 |
| **min** | 0.300000 |
| **25%** | 0.300000 |
| **50%** | 0.470000 |
| **75%** | 0.660000 |
| **max** | 183.770000 |

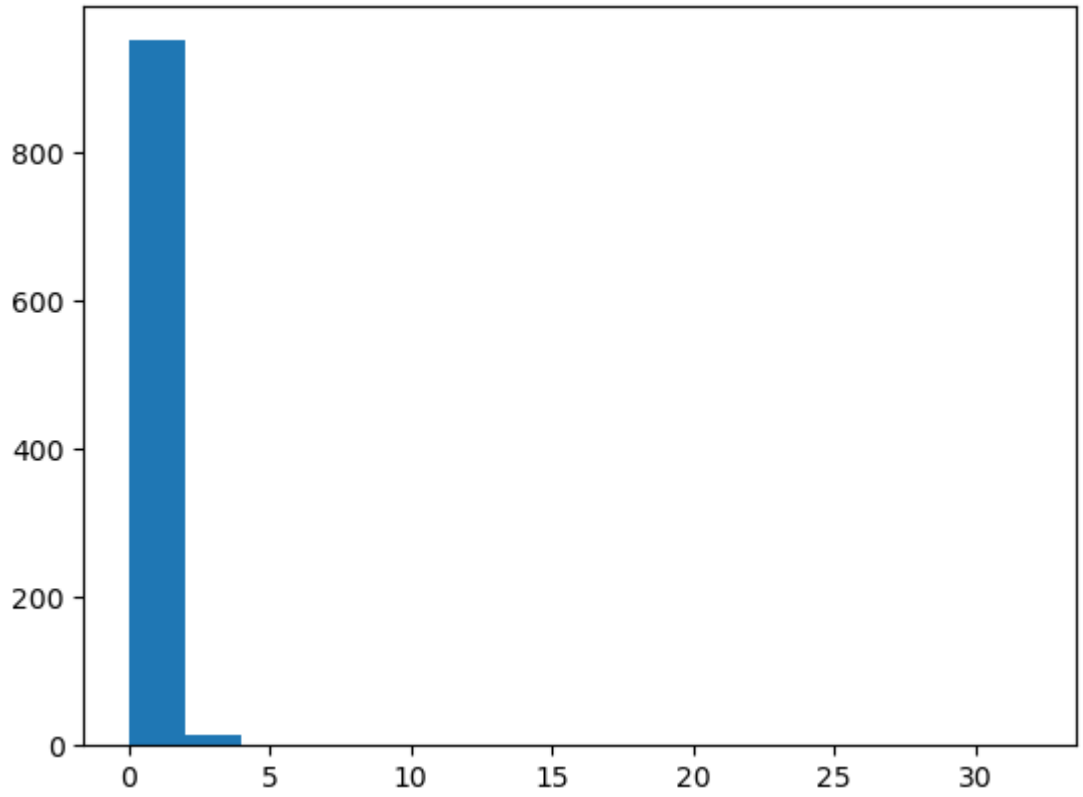
In [36]:
```
plt.hist(eq_diams, bins=np.arange(0,33,2))
```

Out[36]:
```
(array([ 21.,  93., 165., 156., 160., 140.,  85.,  65.,  43.,  29.,  14.,
          7.,   4.,   4.,   4.,   1.]),
 array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 3
2]),
 <BarContainer object of 16 artists>)
```




In [37]:
```
plt.hist(data1['D'], bins=np.arange(0,33,2))
```

Out[37]:
```
(array([951.,  14.,   2.,   2.,   1.,   1.,   1.,   0.,   0.,   0.,   1.,
          0.,   0.,   2.,   1.,   2.]),
 array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 3
2]),
 <BarContainer object of 16 artists>)
```




In [38]:
```
sum(areas)
```

Out[38]:
```
89999.99999999997
```

In [39]:
```
sum(math.pi * data1.D.values * data1.D.values / 4)
```

Out[39]:
```
90046.02431182608
```

In [44]:
```
bigdata1 = data1[data1.D > 10]
```

In [48]:
```
biggestdata1.count()
```

Out[48]:
```
id             3
phi1_degree    3
PHI_degree     3
phi2_degree    3
phi1_rad       3
PHI_rad        3
phi2_rad       3
x              3
y              3
is_edge        3
D              3
dtype: int64
```

In [47]:
```
biggestdata1 = data1[data1.D > 100]
```

# Test functions

In [10]:
```
c = CellComplex.from_tess_file('examples/pass1.tess')
```

In [11]:
```
c
```

Out[11]:
```
<class CellComplex> 2D
1986 vertices
2977 edges
992 faces
```

In [28]:
```
areas = [round(face.area,8) for face in c._faces.values()]
lens = [round(edge.length,8) for edge in c._edges.values()]
```

In [21]:
```
len(areas)
```

Out[21]:
```
992
```

In [22]:
```
len(lens)
```

Out[22]:
```
2977
```

In [29]:
```
areas[:10]
```

Out[29]:
```
[57.65207505,
 8.89500439,
 19.14701803,
 92.32342798,
 25.02556356,
 173.22276588,
 174.86099492,
 103.36229308,
 1.29032029,
 51.6231199]
```

In [24]:
```
lens[:10]
```

Out[24]:
```
[10.55177233945853,
 0.08010325325385241,
 4.33444760600871,
 7.835182623716741,
 0.67399353260125,
 1.826528249735003,
 0.8476322601578334,
 5.346354505155673,
 2.751774260838,
 4.481827883909]
```

In [19]:
```
c.set_measures_from_coo()
```

In [25]:
```
areas_neper = np.loadtxt('/Users/v94623eb2/Documents/ExpDataAnalysis/Experim
```

In [35]:
```
lens_neper = np.loadtxt('/Users/v94623eb2/Documents/ExpDataAnalysis/Experime
```

In [26]:
```
len(areas_neper)
```

Out[26]: 992

In [36]:
```
len(lens_neper)
```

Out[36]: 2977

In [27]:
```
areas_neper[:10]
```

Out[27]:
```
array([ 57.65207505,   8.89500439,  19.14701803,  92.32342798,
        25.02556356, 173.22276588, 174.86099492, 103.36229308,
         1.29032029,  51.6231199 ])
```

In [37]:
```
lens_neper[:10]
```

Out[37]:
```
array([10.55177234,  0.08010325,  4.33444761,  7.83518262,  0.67399353,
        1.82652825,  0.84763226,  5.34635451,  2.75177426,  4.48182788])
```

In [32]:
```
areas[:10]
```

Out[32]:
```
array([ 57.65207505,   8.89500439,  19.14701803,  92.32342798,
        25.02556356, 173.22276588, 174.86099492, 103.36229308,
         1.29032029,  51.6231199 ])
```

In [38]:
```
lens[:10]
```

Out[38]:
```
[10.55177234,
 0.08010325,
 4.33444761,
 7.83518262,
 0.67399353,
 1.82652825,
 0.84763226,
 5.34635451,
 2.75177426,
 4.48182788]
```

In [30]:
```
areas = np.array(areas)
```

In [39]:
```
lens = np.array(lens)
```

In [34]:
```
(areas - areas_neper).min()
```

Out[34]: -4.994999969198943e-09

In [42]:
```
(lens - lens_neper).min()
```

Out[42]: -4.994000102342966e-09

In [47]:
```
c.set_theta_from_ori(lower_thrd=15)
```

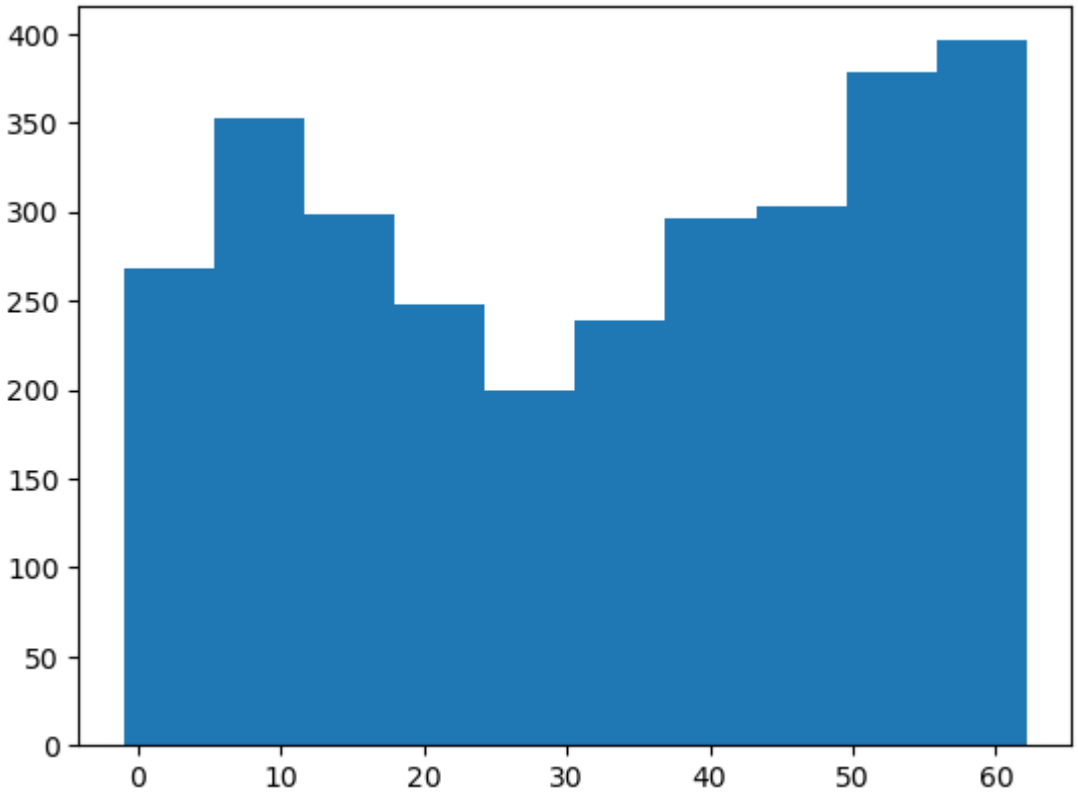
In [48]:
```
special_ids = c.get_special_ids()
```

In [49]:
```
len(special_ids)
```

Out[49]: 2185

In [50]:
```
plt.hist([edge.theta for edge in c.edges])
```

Out[50]:
```
(array([268., 352., 298., 248., 199., 239., 296., 303., 378., 396.]),
 array([-1.        ,  5.31439443, 11.62878886, 17.94318329, 24.25757772,
        30.57197215, 36.88636658, 43.20076101, 49.51515543, 55.82954986,
        62.14394429]),
 <BarContainer object of 10 artists>)
```




In [56]:
```
pd.DataFrame([(1,2), (3, 4), (5,6)], columns=['x', 'y']).to_dict('list')
```

Out[56]:
```
{'x': [1, 3, 5], 'y': [2, 4, 6]}
```

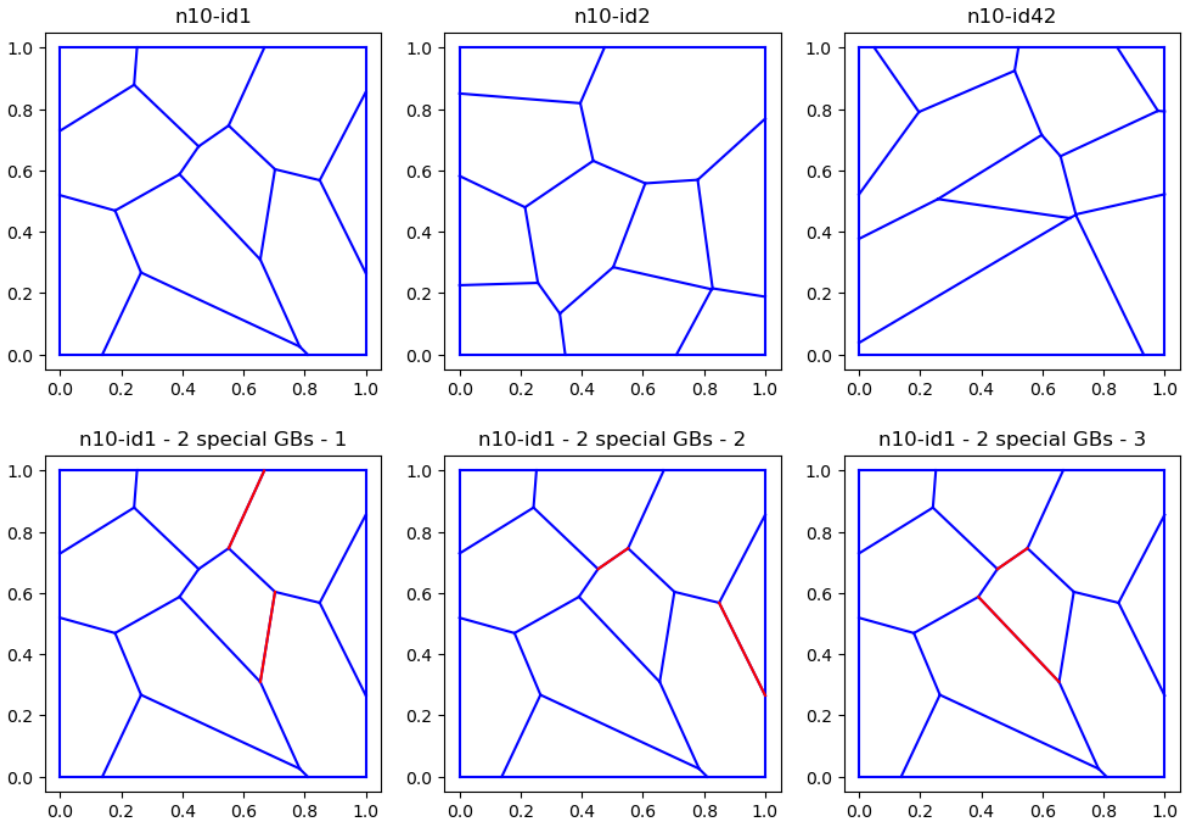
In [55]:
```
dict(x=[1,3,5], y=[2,4,6])
```

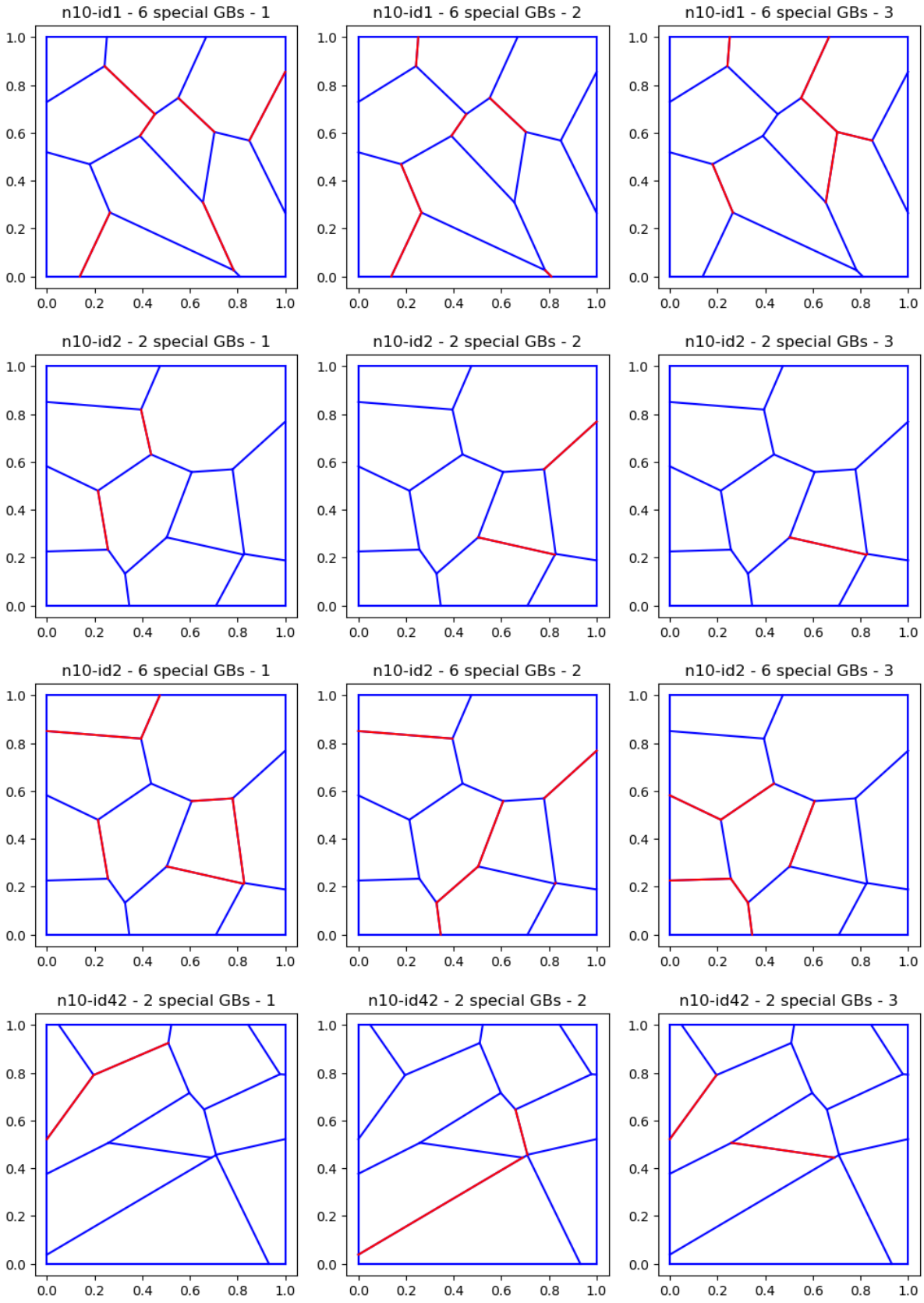
Out[55]:
```
{'x': [1, 3, 5], 'y': [2, 4, 6]}
```

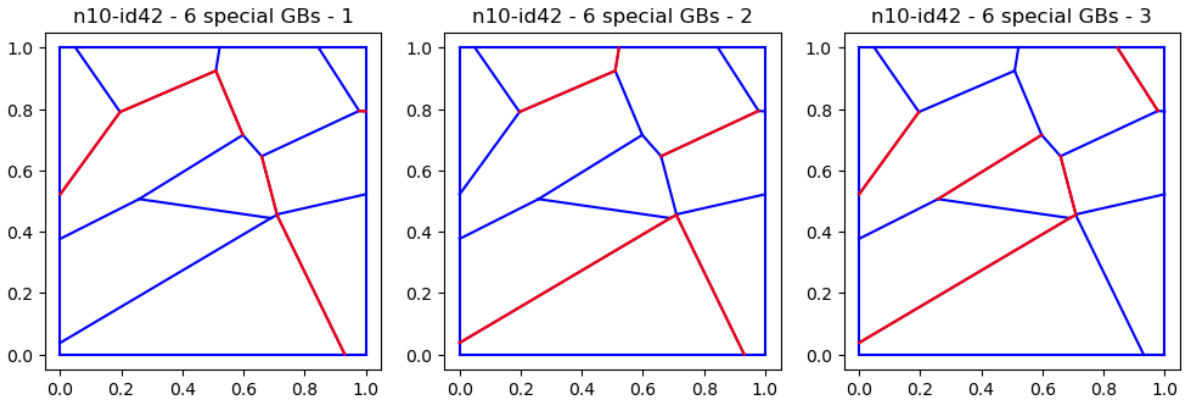
# Model 2D

In [71]:
```
n = 10
n_spec_list = [2, 6]

fig, axes = plt.subplots(1, 3, subplot_kw={'aspect': 1}, figsize=(10,10))

for i, id_ in enumerate([1, 2, 42], 1):
    filename = f'sim_examples/n{n}-id{id_}.tess'
    c = CellComplex.from_tess_file(filename)
```

```
    c.plot_edges(color='blue', ax=axes[i - 1])
    axes[i - 1].set_title(f'n{n}-id{id_}')

plt.tight_layout()
plt.show()
plt.close()

for id_ in [1, 2, 42]:
    for n_spec in n_spec_list:

        filename = f'sim_examples/n{n}-id{id_}.tess'
        c = CellComplex.from_tess_file(filename)

        fig, axes = plt.subplots(1, 3, subplot_kw={'aspect': 1}, figsize=(10

        for i in [1, 2, 3]:
            special_filename = f'sim_examples/n{n}-id{id_}-spec-{n_spec}-{i}
            special_ids = np.loadtxt(special_filename)
            c.reset_special(special_ids=special_ids)
            c.plot_edges(color='blue', ax=axes[i - 1])
            c.plot_edges(c.get_special_ids(), color='red', ax=axes[i - 1])
            axes[i - 1].set_title(f'n{n}-id{id_} - {n_spec} special GBs - {i

        plt.tight_layout()
        plt.show()
        plt.close()
```

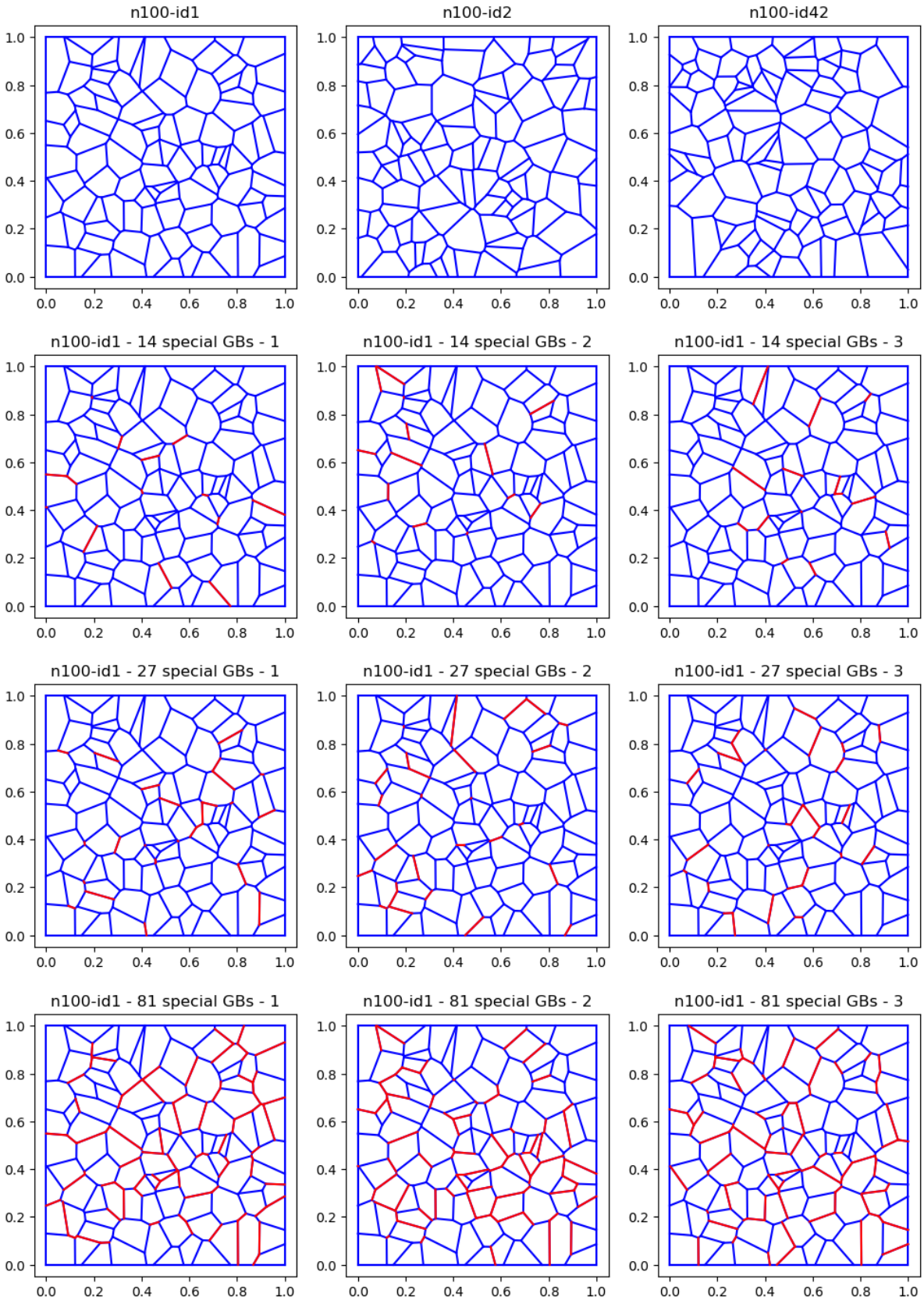


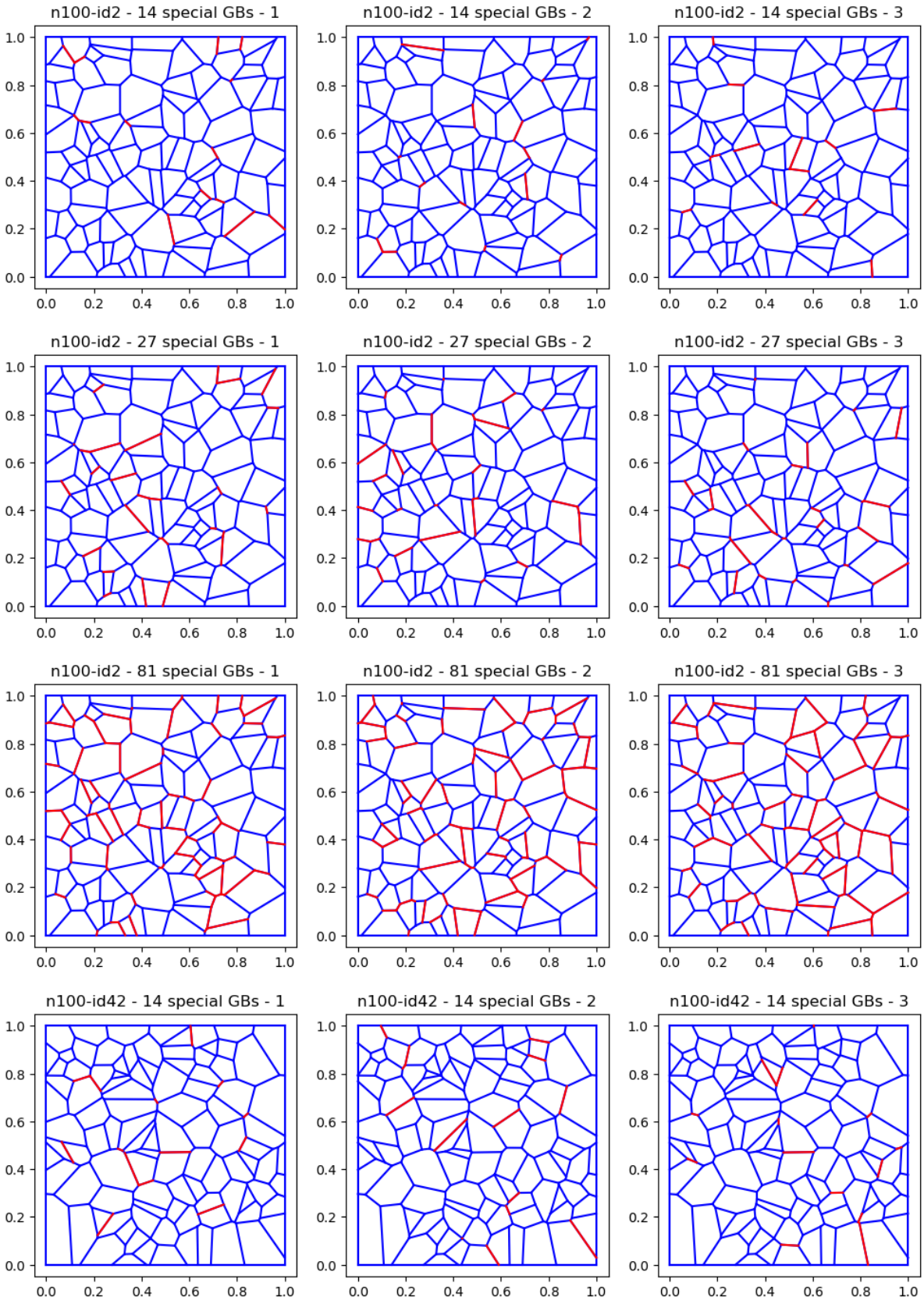

n10-id1 - 6 special GBs - 1
n10-id1 - 6 special GBs - 2
n10-id1 - 6 special GBs - 3
n10-id2 - 2 special GBs - 1
n10-id2 - 2 special GBs - 2
n10-id2 - 2 special GBs - 3
n10-id2 - 6 special GBs - 1
n10-id2 - 6 special GBs - 2
n10-id2 - 6 special GBs - 3
n10-id42 - 2 special GBs - 1
n10-id42 - 2 special GBs - 2
n10-id42 - 2 special GBs - 3
0.0 0.2 0.4 0.6 0.8 1.0

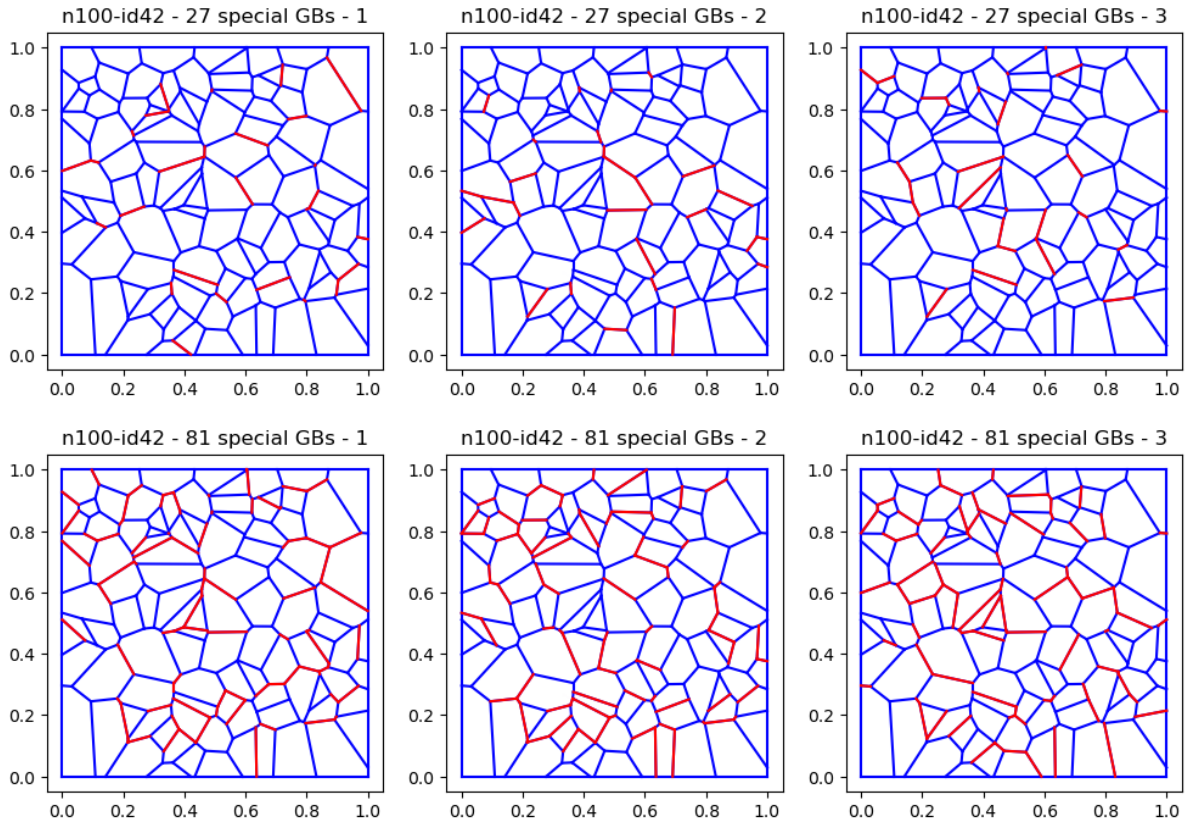



In [72]:

```
n = 100
n_spec_list = [14, 27, 81] # 5%, 10%, 30%

fig, axes = plt.subplots(1, 3, subplot_kw={'aspect': 1}, figsize=(10,10))

for i, id_ in enumerate([1, 2, 42], 1):
    filename = f'sim_examples/n{n}-id{id_}.tess'
    c = CellComplex.from_tess_file(filename)
    c.plot_edges(color='blue', ax=axes[i - 1])
    axes[i - 1].set_title(f'n{n}-id{id_}')

plt.tight_layout()
plt.show()
plt.close()

for id_ in [1, 2, 42]:
    for n_spec in n_spec_list:

        filename = f'sim_examples/n{n}-id{id_}.tess'
        c = CellComplex.from_tess_file(filename)

        fig, axes = plt.subplots(1, 3, subplot_kw={'aspect': 1}, figsize=(10

        for i in [1, 2, 3]:
            special_filename = f'sim_examples/n{n}-id{id_}-spec-{n_spec}-{i}
            special_ids = np.loadtxt(special_filename)
            c.reset_special(special_ids=special_ids)
            c.plot_edges(color='blue', ax=axes[i - 1])
            c.plot_edges(c.get_special_ids(), color='red', ax=axes[i - 1])
            axes[i - 1].set_title(f'n{n}-id{id_} - {n_spec} special GBs - {i

        plt.tight_layout()
        plt.show()
        plt.close()
```


n100-id1
n100-id2
n100-id42
n100-id1 - 14 special GBs - 1
n100-id1 - 14 special GBs - 2
n100-id1 - 14 special GBs - 3
n100-id1 - 27 special GBs - 1
n100-id1 - 27 special GBs - 2
n100-id1 - 27 special GBs - 3
n100-id1 - 81 special GBs - 1
n100-id1 - 81 special GBs - 2
n100-id1 - 81 special GBs - 3
0.0
0.2
0.4
0.6
0.8
1.0

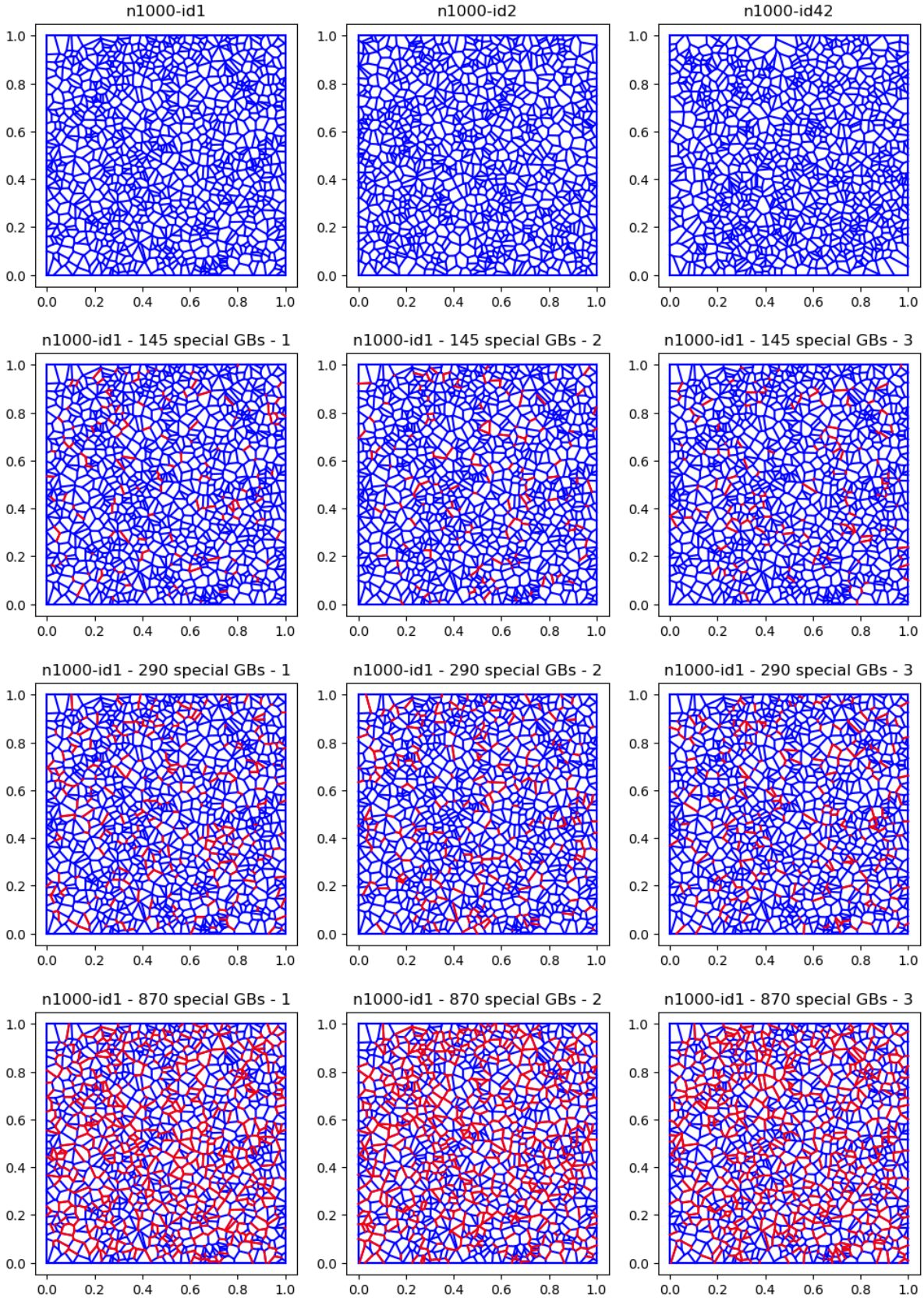

n100-id2 - 14 special GBs - 1
n100-id2 - 14 special GBs - 2
n100-id2 - 14 special GBs - 3
n100-id2 - 27 special GBs - 1
n100-id2 - 27 special GBs - 2
n100-id2 - 27 special GBs - 3
n100-id2 - 81 special GBs - 1
n100-id2 - 81 special GBs - 2
n100-id2 - 81 special GBs - 3
n100-id42 - 14 special GBs - 1
n100-id42 - 14 special GBs - 2
n100-id42 - 14 special GBs - 3
0.0
0.2
0.4
0.6
0.8
1.0

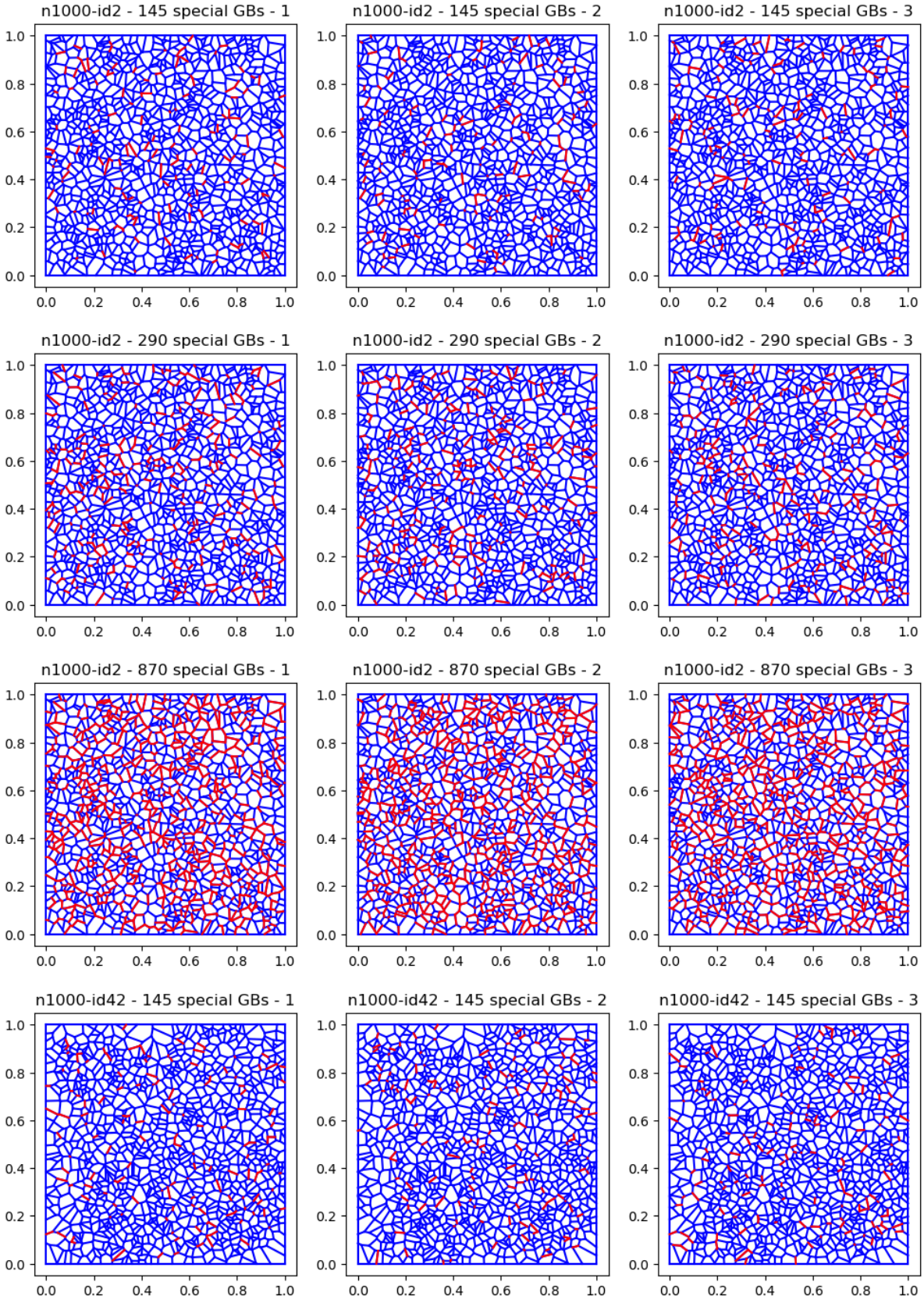


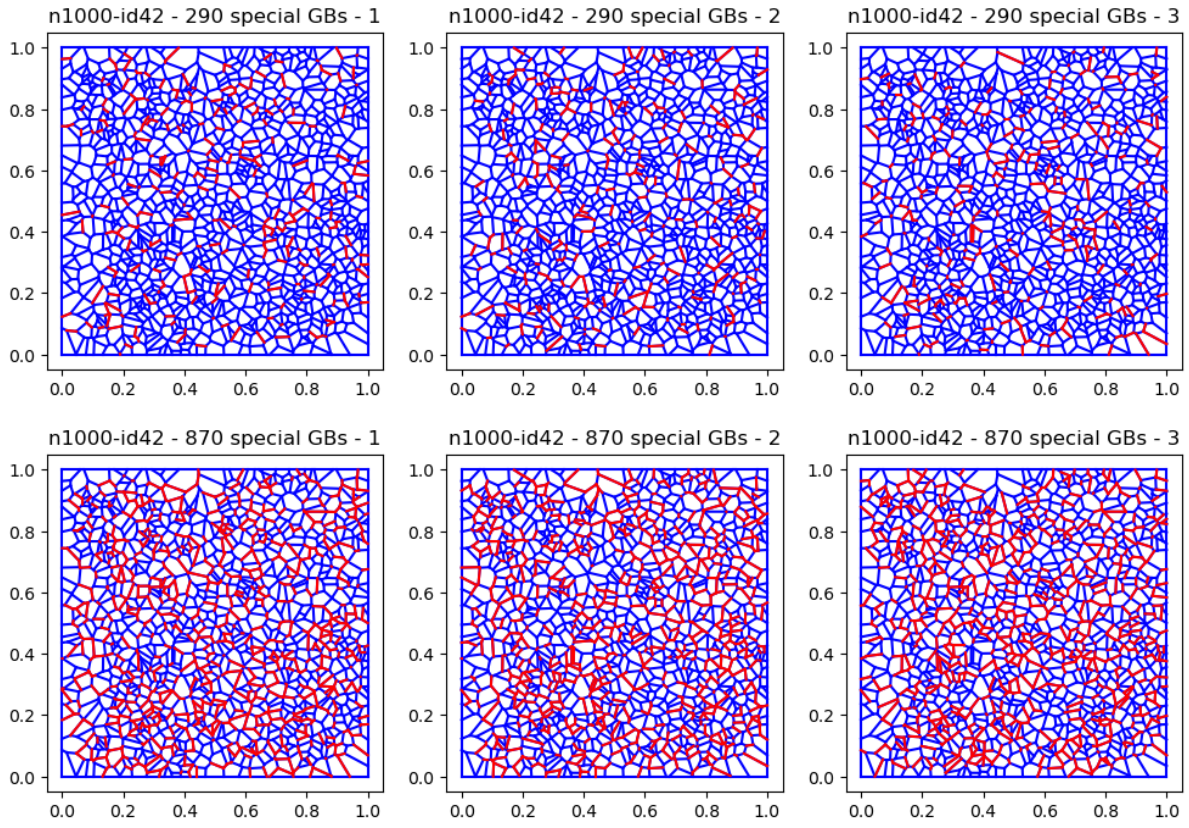

In [75]:

```
n = 1000
n_spec_list = [145, 290, 870] # 5%, 10%, 30%

fig, axes = plt.subplots(1, 3, subplot_kw={'aspect': 1}, figsize=(10,10))

for i, id_ in enumerate([1, 2, 42], 1):
    filename = f'sim_examples/n{n}-id{id_}.tess'
    c = CellComplex.from_tess_file(filename)
    c.plot_edges(color='blue', ax=axes[i - 1])
    axes[i - 1].set_title(f'n{n}-id{id_}')

plt.tight_layout()
plt.show()
plt.close()

for id_ in [1, 2, 42]:
    for n_spec in n_spec_list:

        filename = f'sim_examples/n{n}-id{id_}.tess'
        c = CellComplex.from_tess_file(filename)

        fig, axes = plt.subplots(1, 3, subplot_kw={'aspect': 1}, figsize=(10

        for i in [1, 2, 3]:
            special_filename = f'sim_examples/n{n}-id{id_}-spec-{n_spec}-{i}
            special_ids = np.loadtxt(special_filename)
            c.reset_special(special_ids=special_ids)
            c.plot_edges(color='blue', ax=axes[i - 1])
            c.plot_edges(c.get_special_ids(), color='red', ax=axes[i - 1])
            axes[i - 1].set_title(f'n{n}-id{id_} - {n_spec} special GBs - {i
```

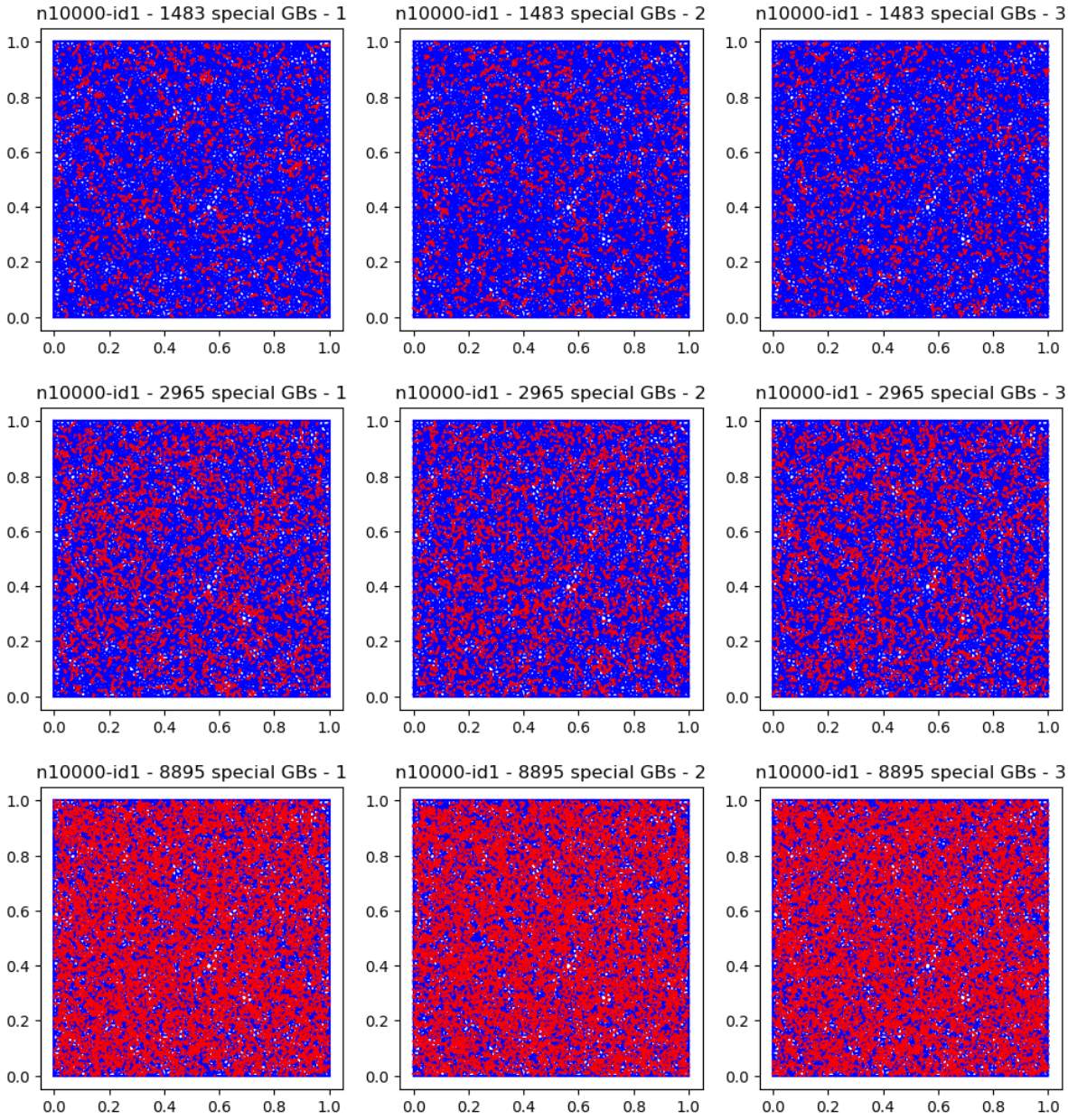
```
    plt.tight_layout()
    plt.show()
    plt.close()
```




n1000-id2 - 145 special GBs - 1
n1000-id2 - 145 special GBs - 2
n1000-id2 - 145 special GBs - 3
n1000-id2 - 290 special GBs - 1
n1000-id2 - 290 special GBs - 2
n1000-id2 - 290 special GBs - 3
n1000-id2 - 870 special GBs - 1
n1000-id2 - 870 special GBs - 2
n1000-id2 - 870 special GBs - 3
n1000-id42 - 145 special GBs - 1
n1000-id42 - 145 special GBs - 2
n1000-id42 - 145 special GBs - 3
0.0
0.2
0.4
0.6
0.8
1.0

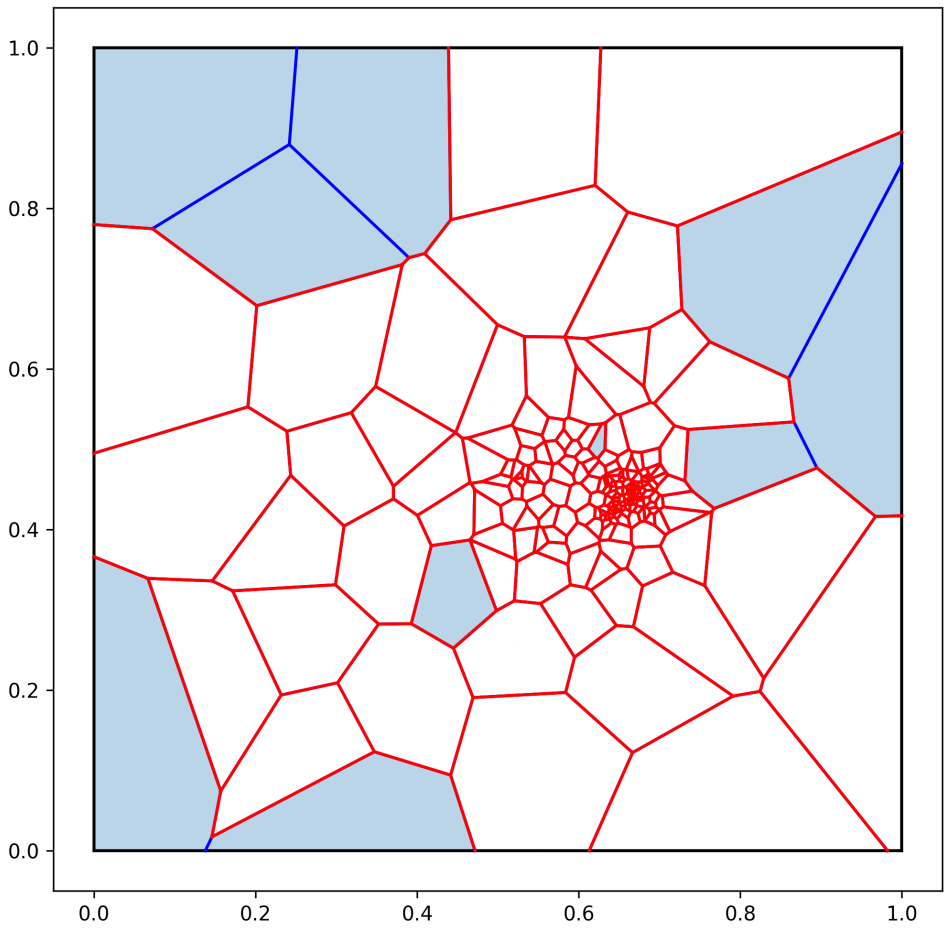



In [74]:
```
n = 10000
id_ = 1
n_spec_list = [1483, 2965, 8895] # 5%, 10%, 30%

for n_spec in n_spec_list:

    filename = f'sim_examples/n{n}-id{id_}.tess'
    c = CellComplex.from_tess_file(filename)

    fig, axes = plt.subplots(1, 3, subplot_kw={'aspect': 1}, figsize=(10,10)

    for i in [1, 2, 3]:
        special_filename = f'sim_examples/n{n}-id{id_}-spec-{n_spec}-{i}.txt
        special_ids = np.loadtxt(special_filename)
        c.reset_special(special_ids=special_ids)
        c.plot_edges(color='blue', ax=axes[i - 1])
        c.plot_edges(c.get_special_ids(), color='red', ax=axes[i - 1])
        axes[i - 1].set_title(f'n{n}-id{id_} - {n_spec} special GBs - {i}')

    plt.tight_layout()
    plt.show()
    plt.close()
```

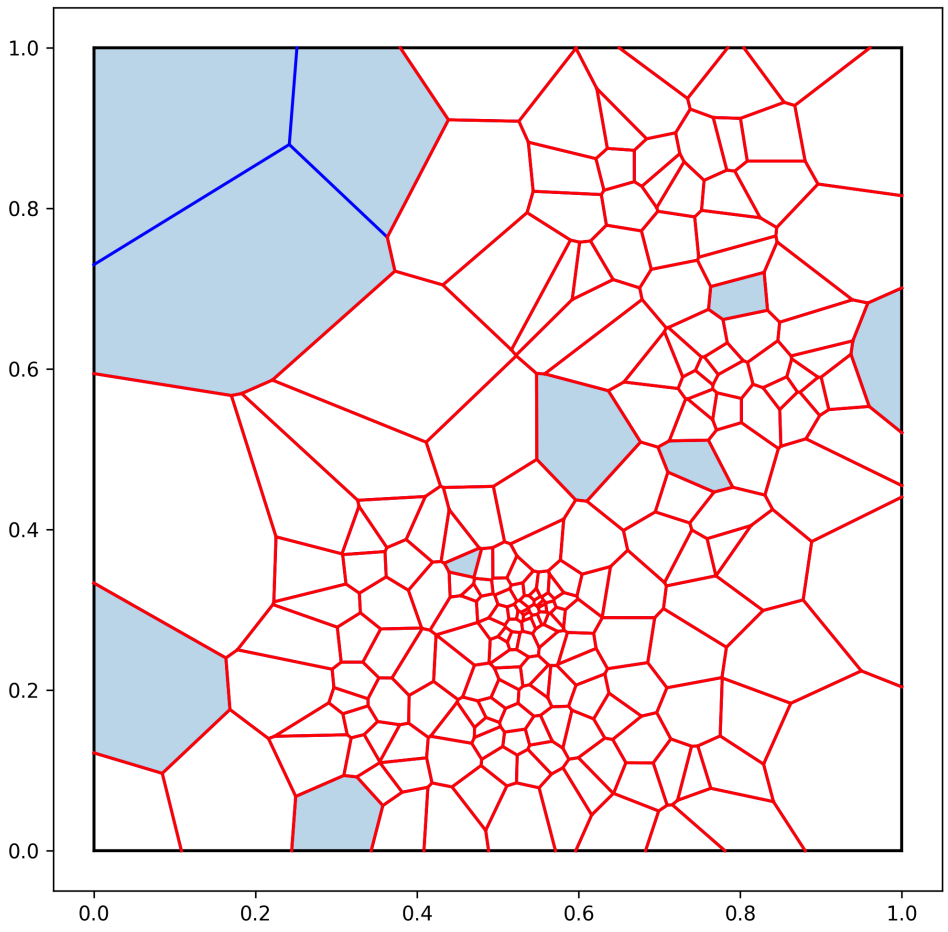



# Results

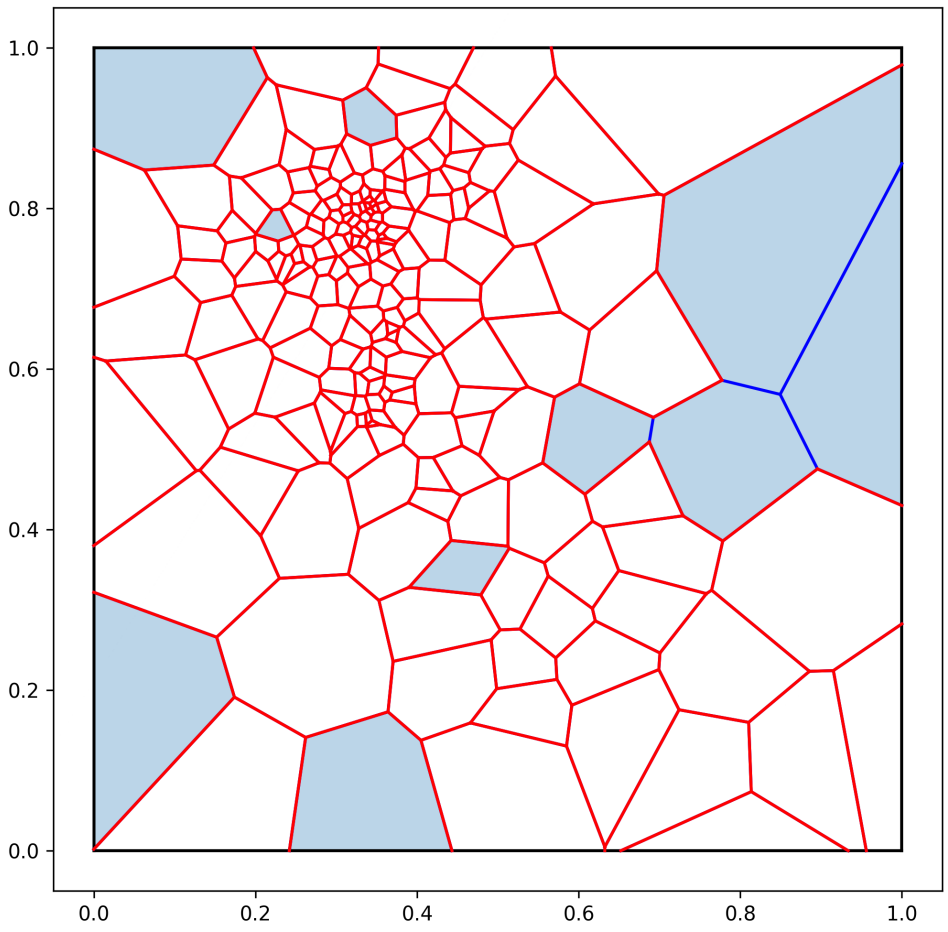
n = 10, id = 1, n_spec = 2-1, N_new_grains = 200




n = 10, id = 1, n_spec = 2-2, N_new_grains = 200

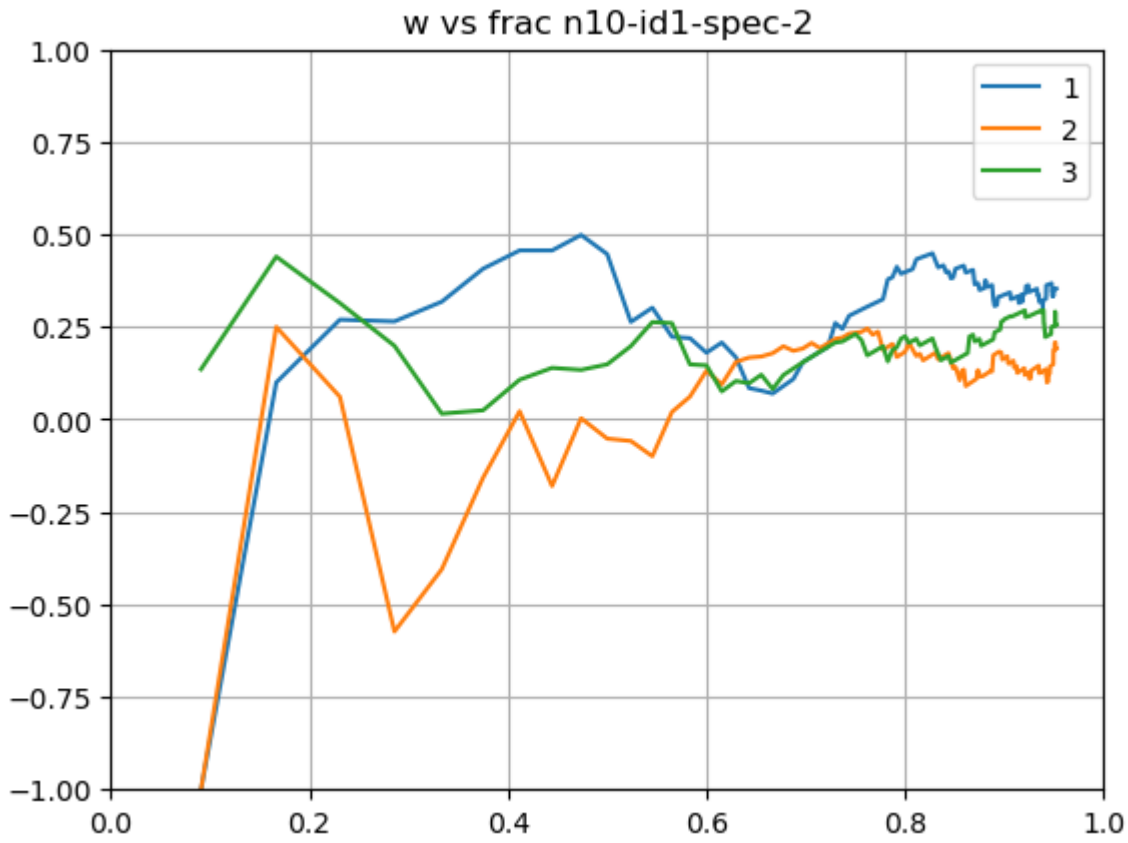



n = 10, id = 1, n_spec = 2-3, N_new_grains = 200

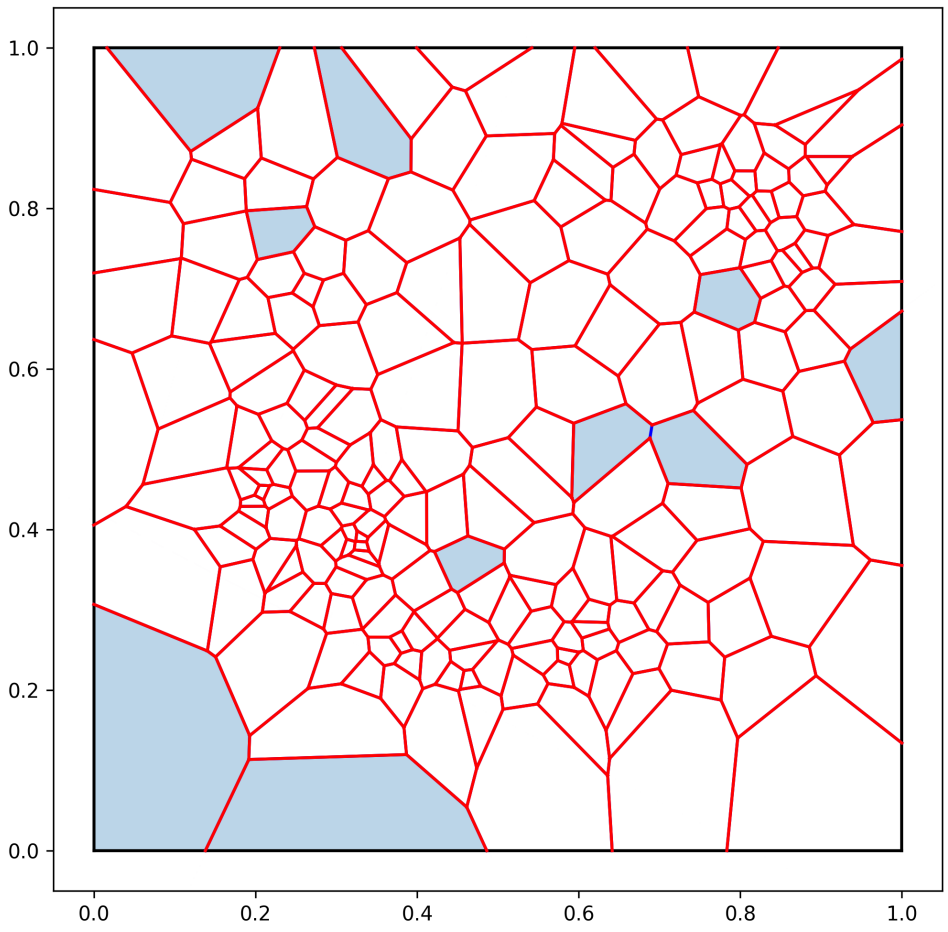



In [83]:

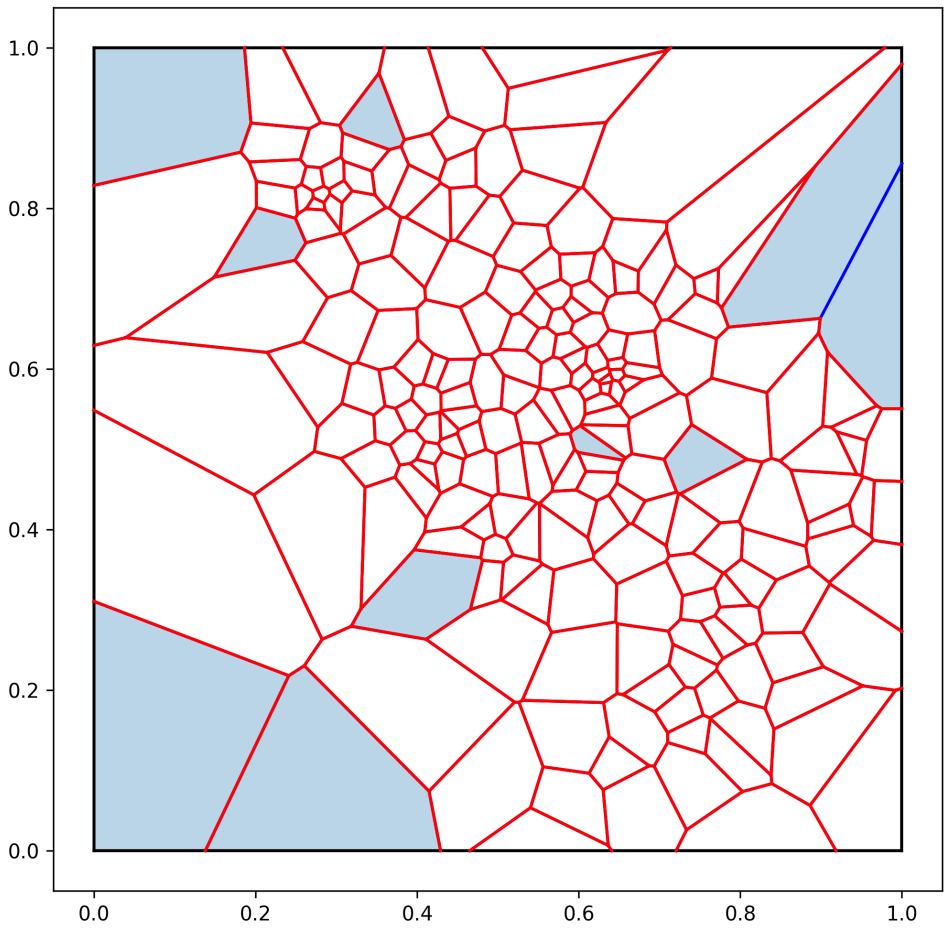
```
n = 10
id_ = 1
n_spec = 2 # 10%

fig, ax = plt.subplots(1, 1)
for i in [1, 2, 3]:
    results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{20*n}-1/resul
    ax.plot(results['frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs frac n{n}-id{id_}-spec-{n_spec}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()
```



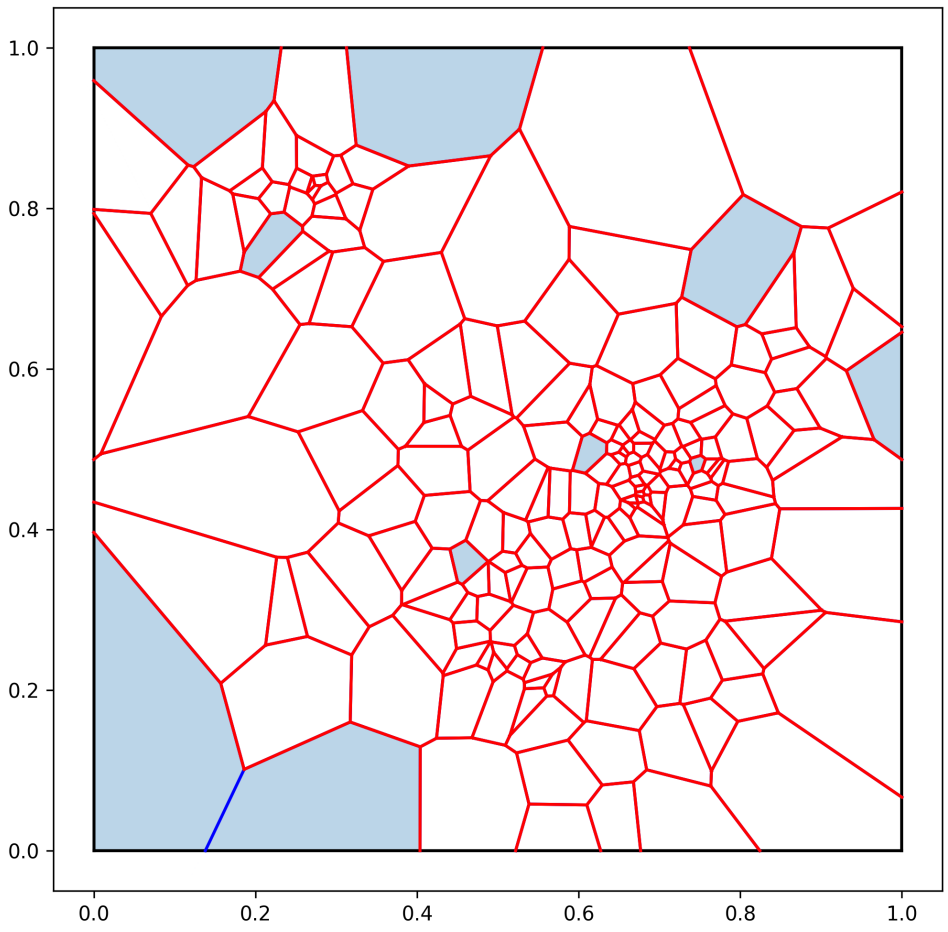

n = 10, id = 1, n_spec = 6-1, N_new_grains = 200




n = 10, id = 1, n_spec = 6-2, N_new_grains = 200



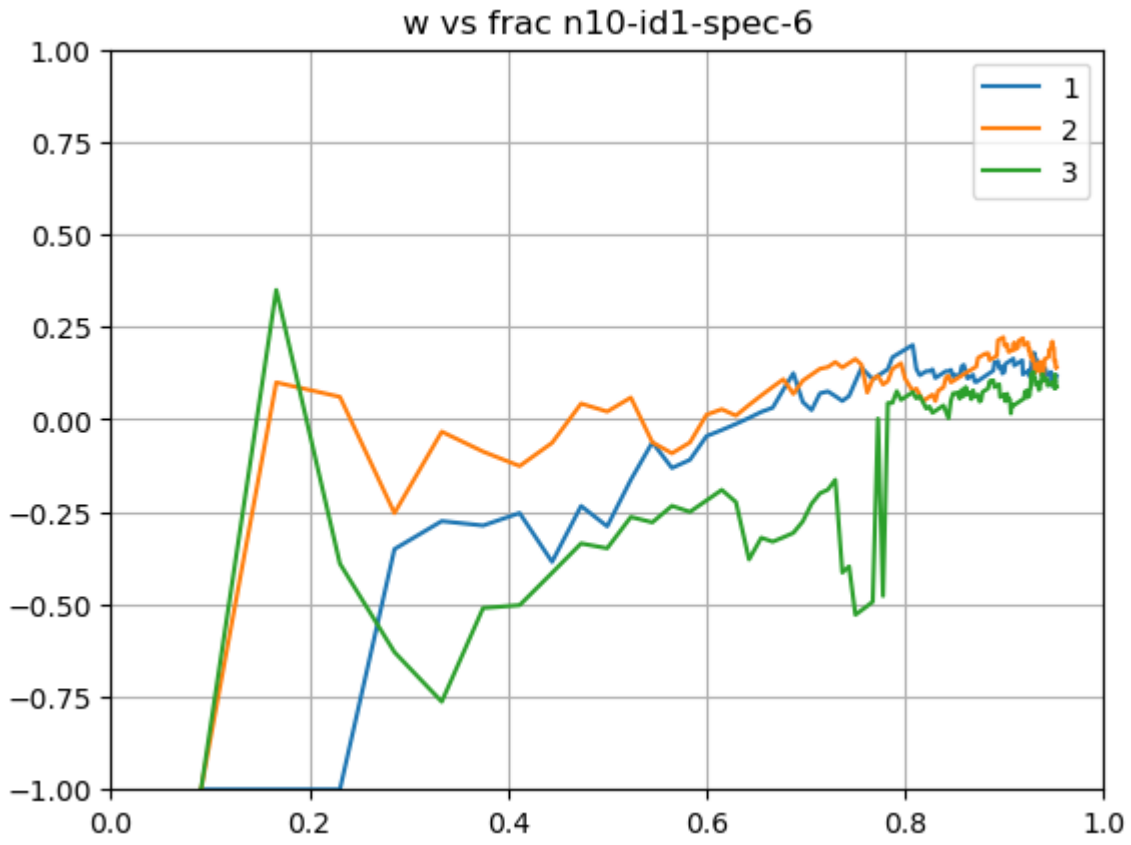

n = 10, id = 1, n_spec = 6-3, N_new_grains = 200

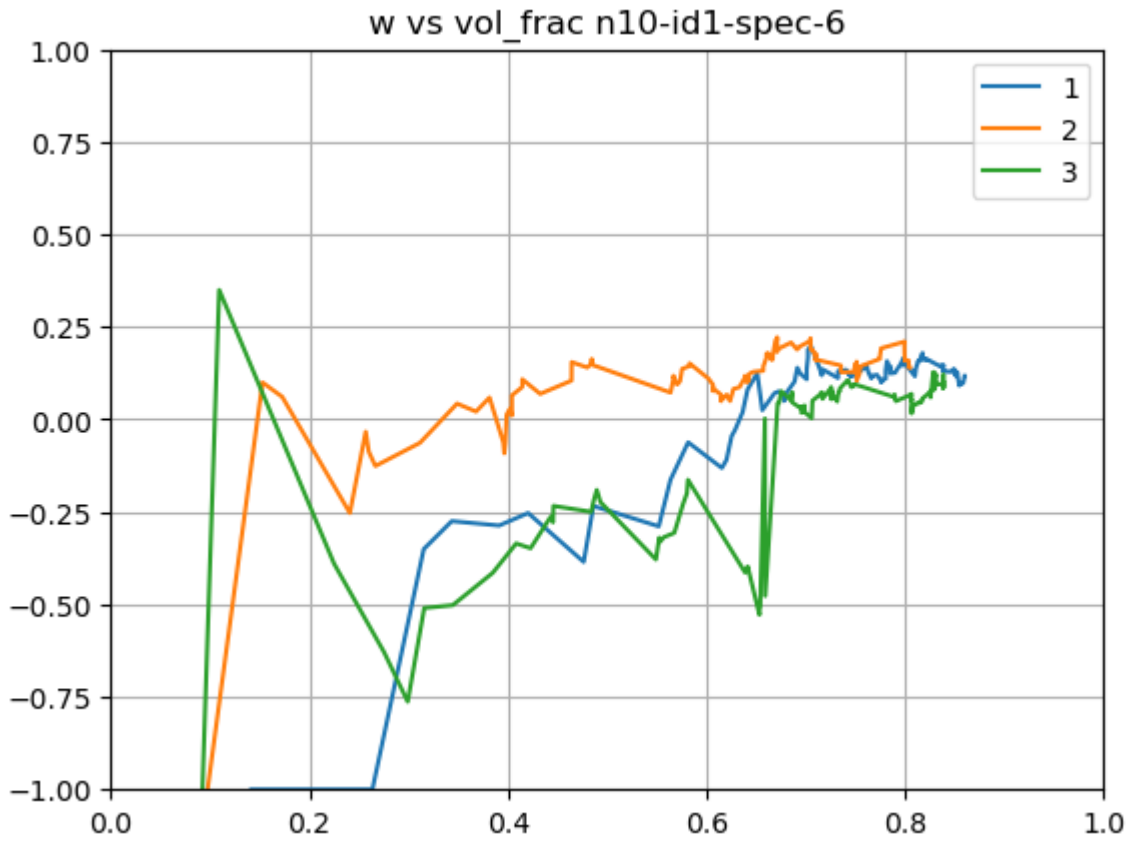



In [86]:

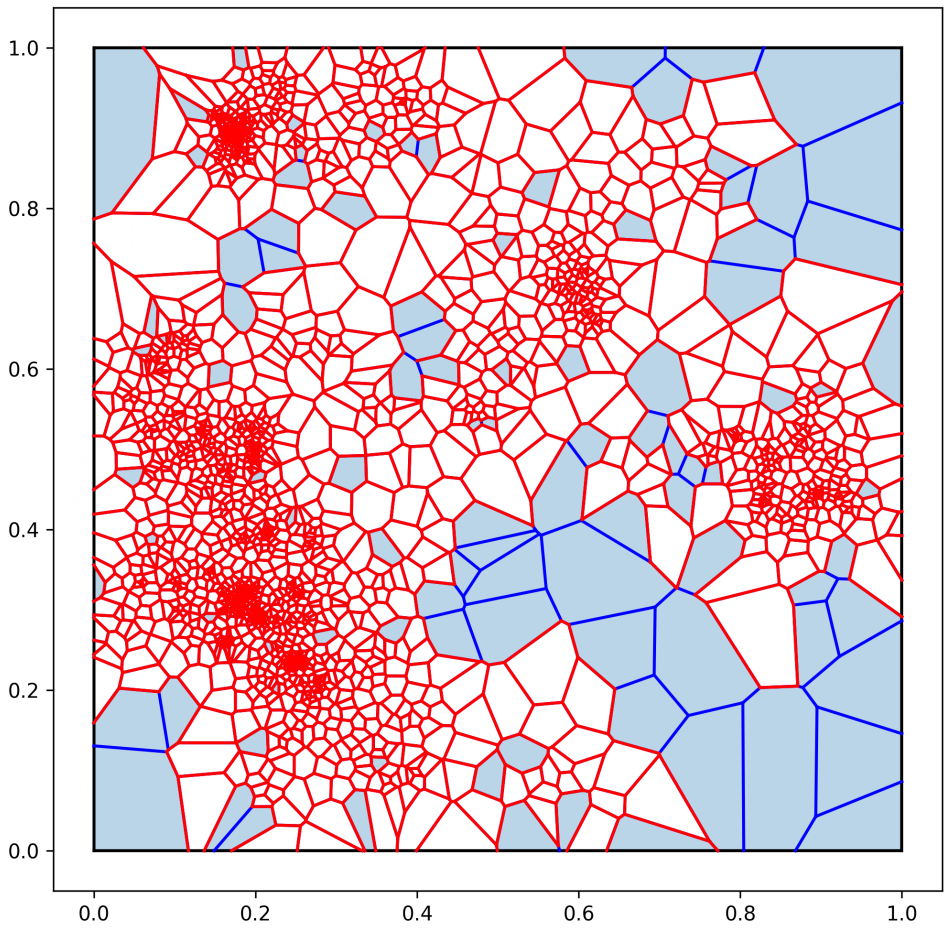
```
n = 10
id_ = 1
n_spec = 6 # 30%

fig, ax = plt.subplots(1, 1)
for i in [1, 2, 3]:
    results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{20*n}-1/resul
    ax.plot(results['frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs frac n{n}-id{id_}-spec-{n_spec}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()

fig, ax = plt.subplots(1, 1)
for i in [1, 2, 3]:
    results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{20*n}-1/resul
    ax.plot(results['vol_frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs vol_frac n{n}-id{id_}-spec-{n_spec}')
```

```
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()
```

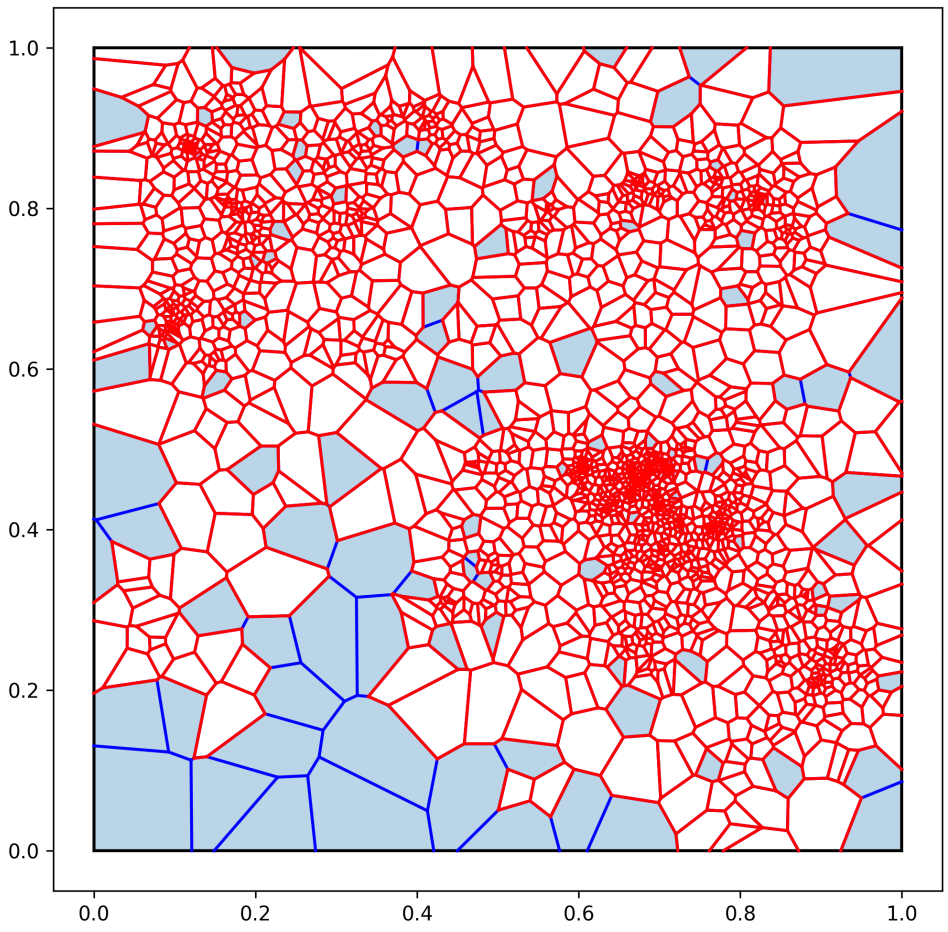




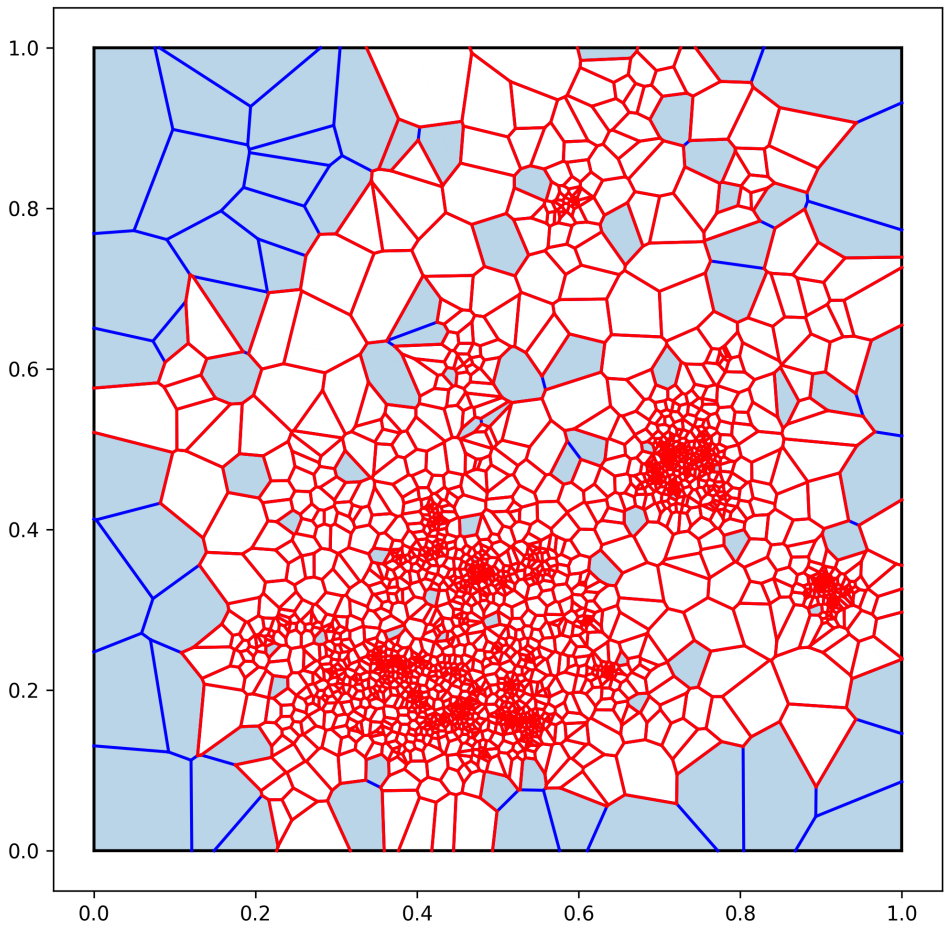

n = 100, id = 1, spec = 14-1, N_new_grains = 2000




n = 100, id = 1, spec = 14-2, N_new_grains = 2000



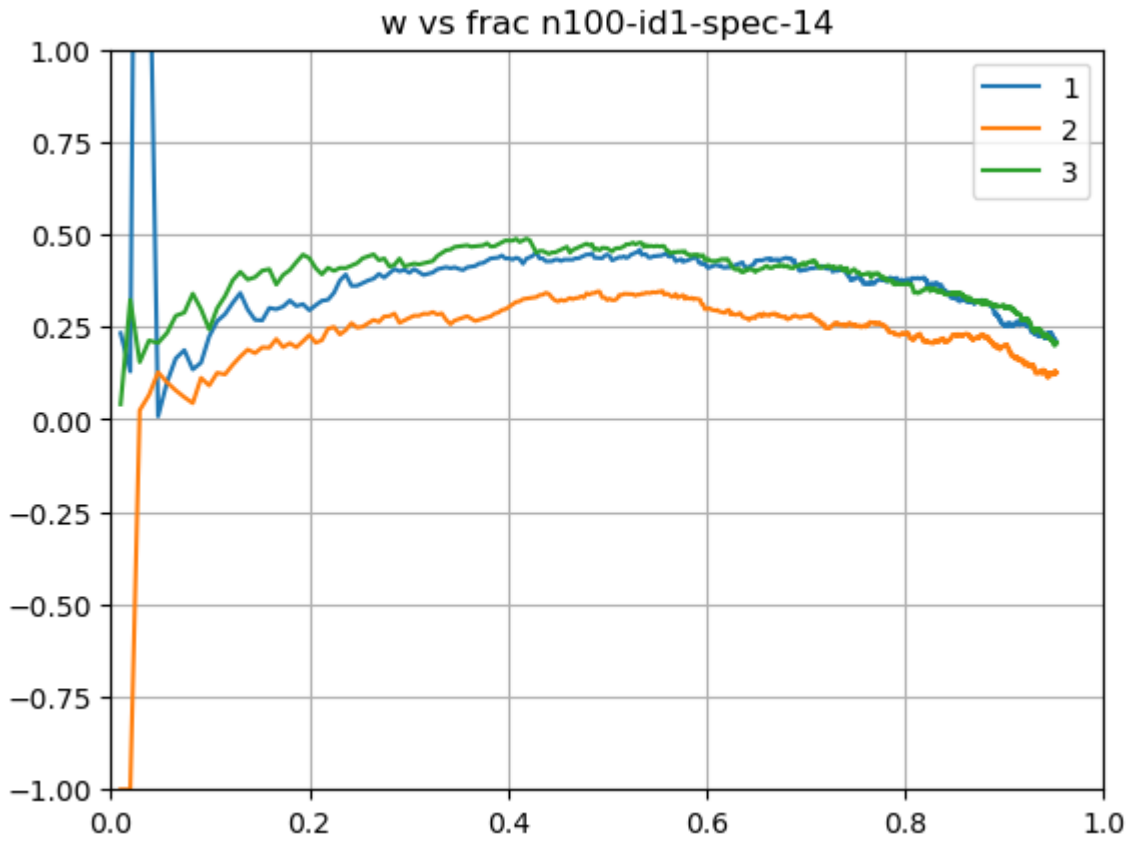

n = 100, id = 1, spec = 14-3, N_new_grains = 2000

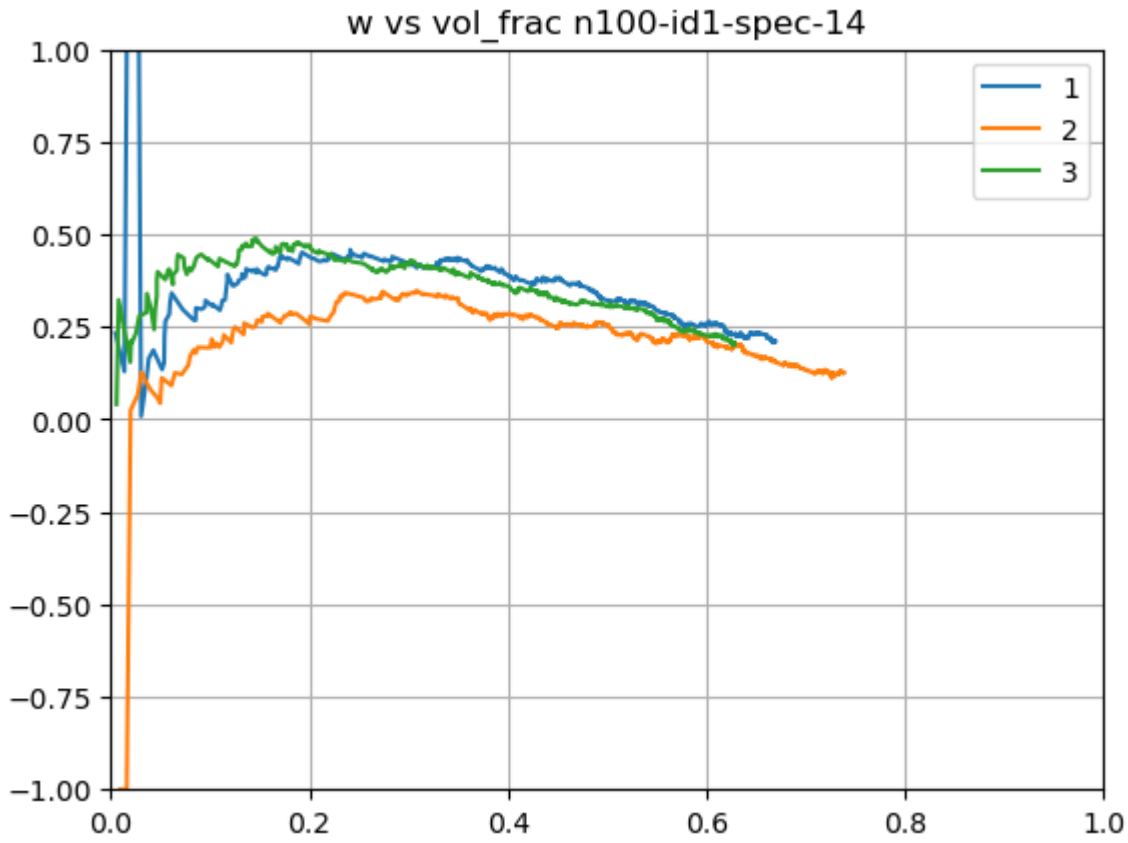



In [88]:

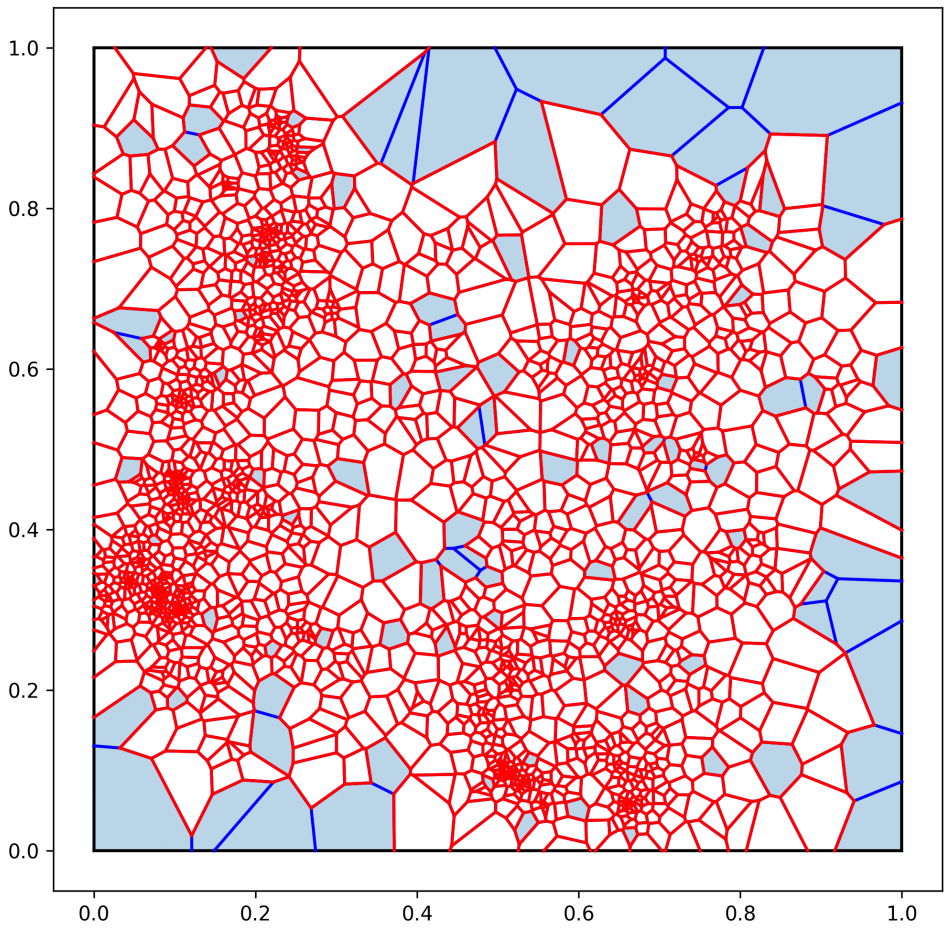
```
n = 100
id_ = 1
n_spec_list = 14 # 5%

fig, ax = plt.subplots(1, 1)
for i in [1,2,3]:
    results = pd.read_csv(f'results/n{n}-id{id_}-{k}-{i}-{20*n}-1/results.tx

    ax.plot(results['frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs frac n{n}-id{id_}-spec-{k}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()

fig, ax = plt.subplots(1, 1)
for i in [1,2,3]:
    results = pd.read_csv(f'results/n{n}-id{id_}-{k}-{i}-{20*n}-1/results.tx
```

```
    ax.plot(results['vol_frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs vol_frac n{n}-id{id_}-spec-{k}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()
```

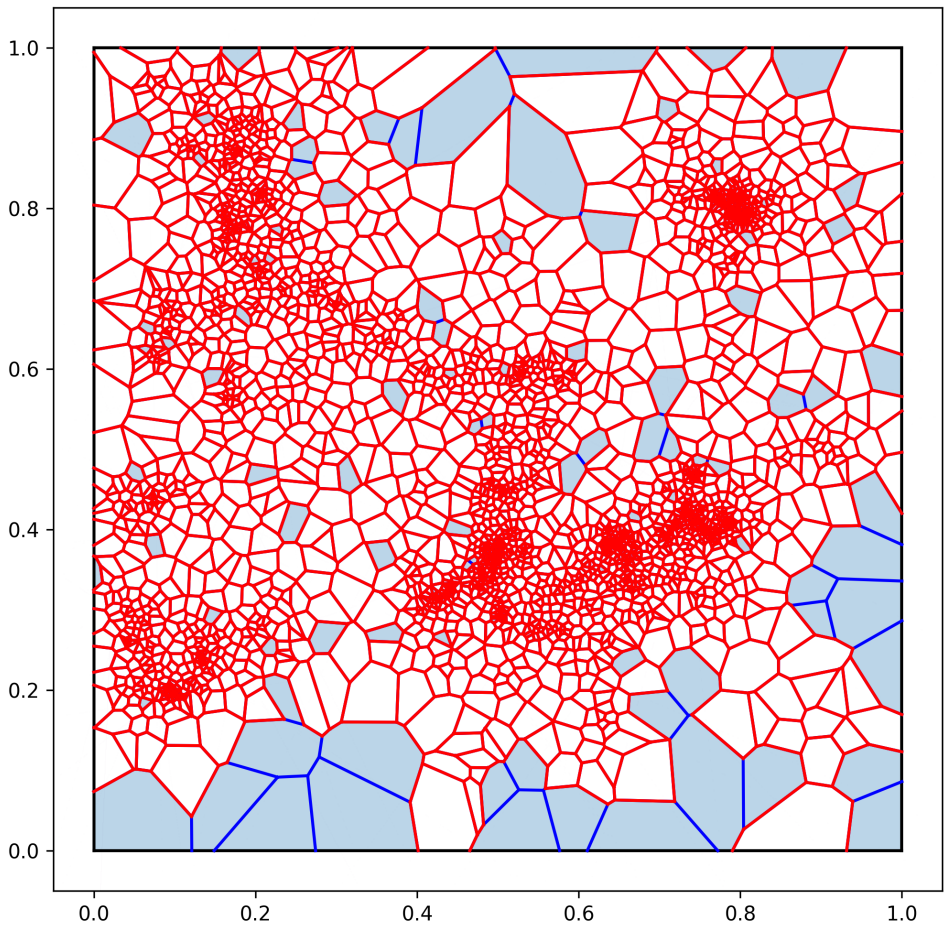




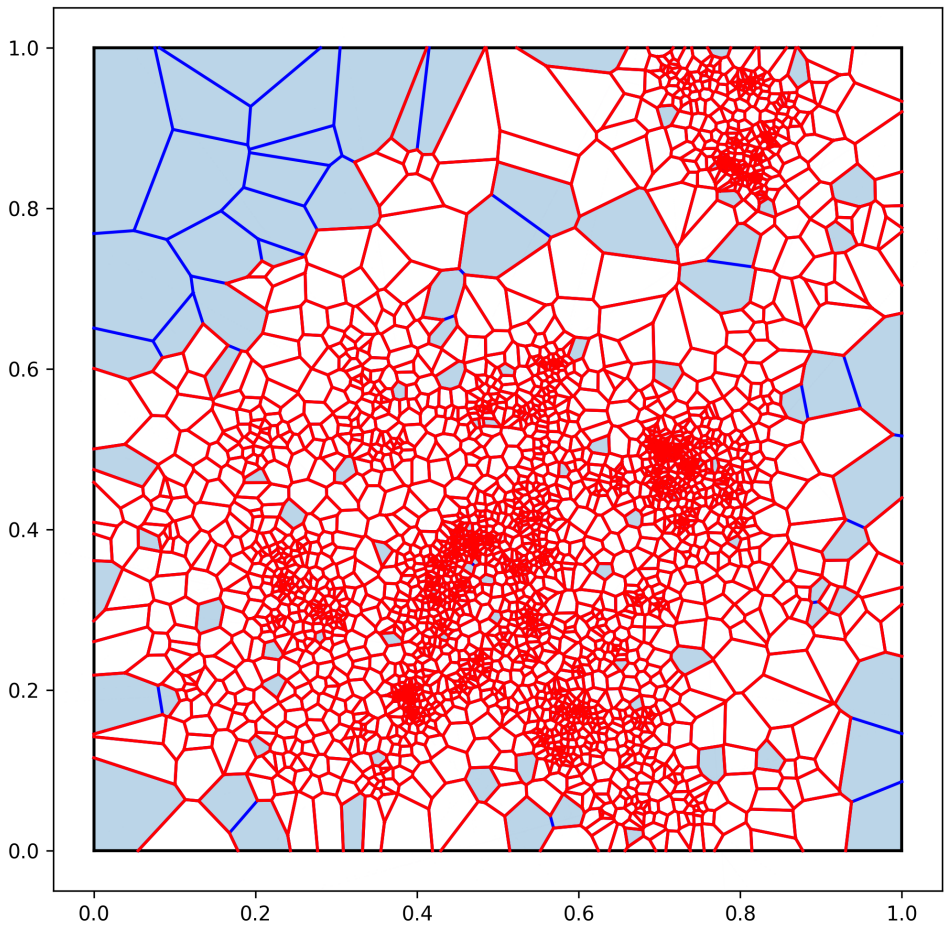

n = 100, id = 1, spec = 14-1, N_new_grains = 2000 (200 steps with 10 new)




n = 100, id = 1, spec = 14-2, N_new_grains = 3000 (300 steps with 10 new)



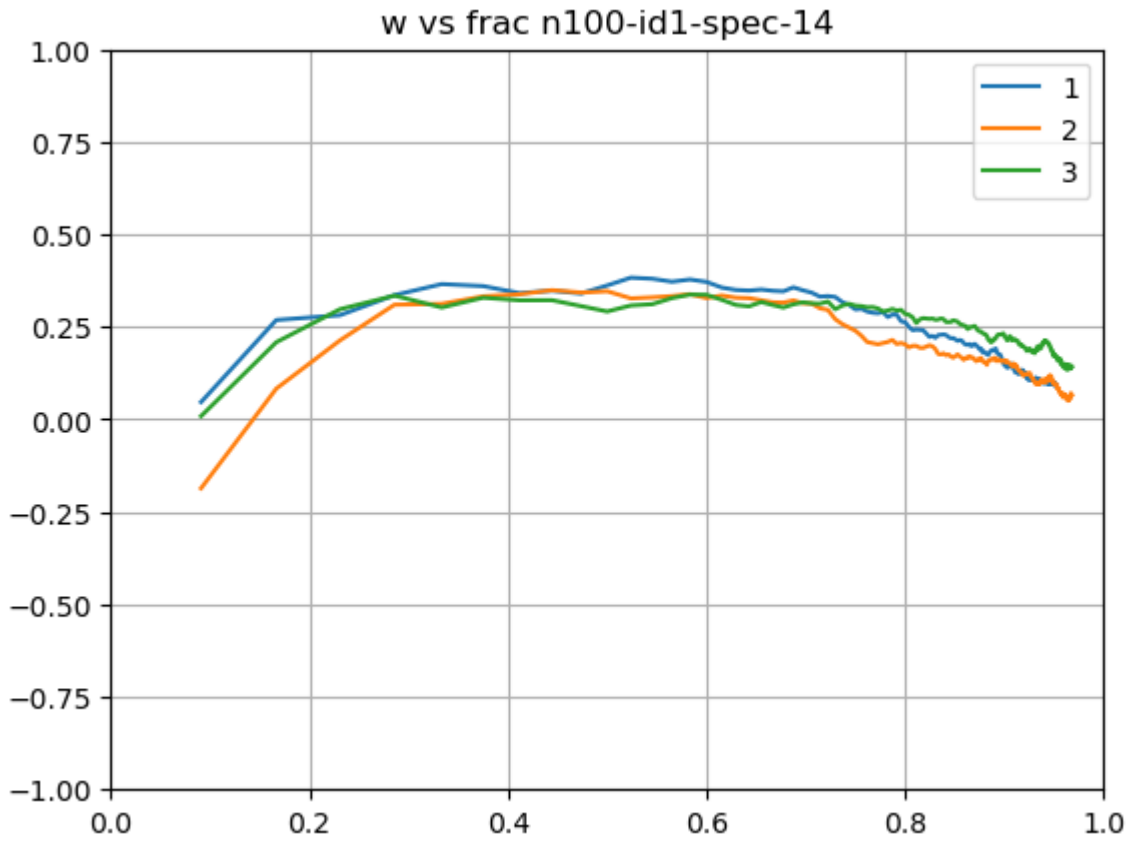

n = 100, id = 1, spec = 14-3, N_new_grains = 3000 (300 steps with 10 new)

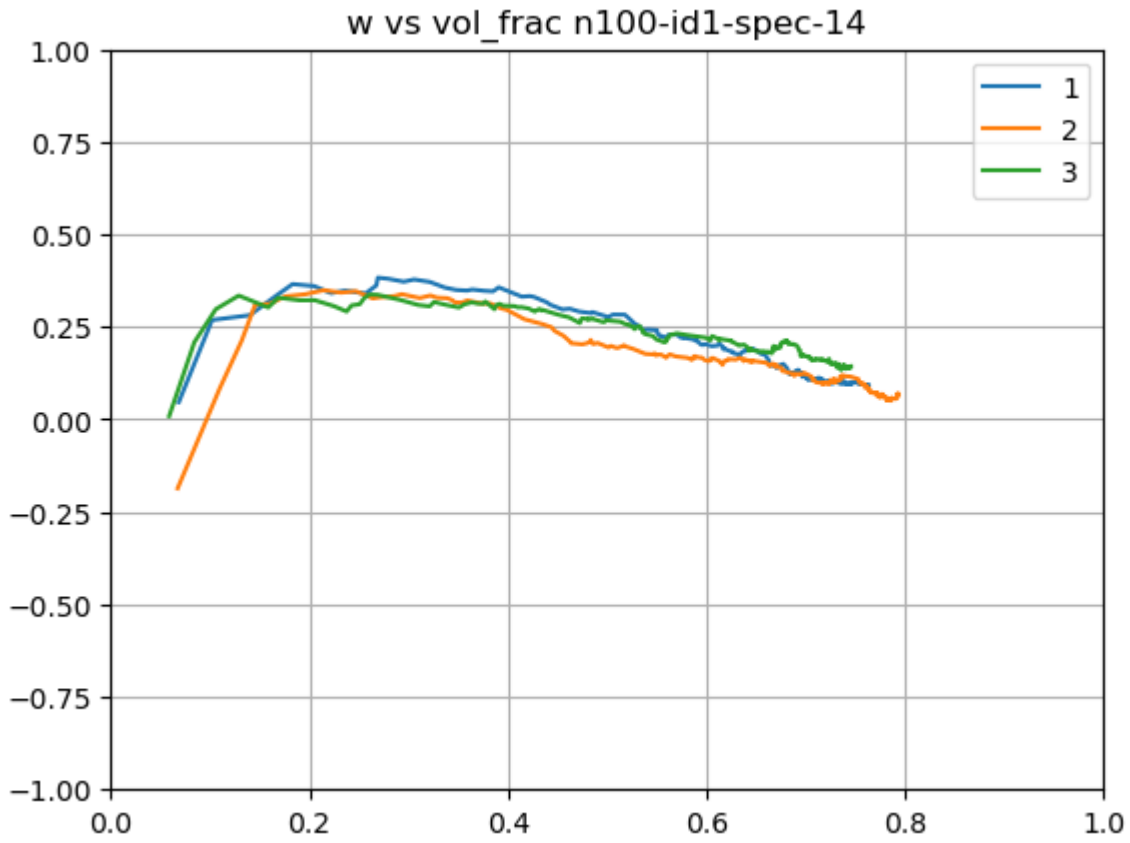



In [99]:

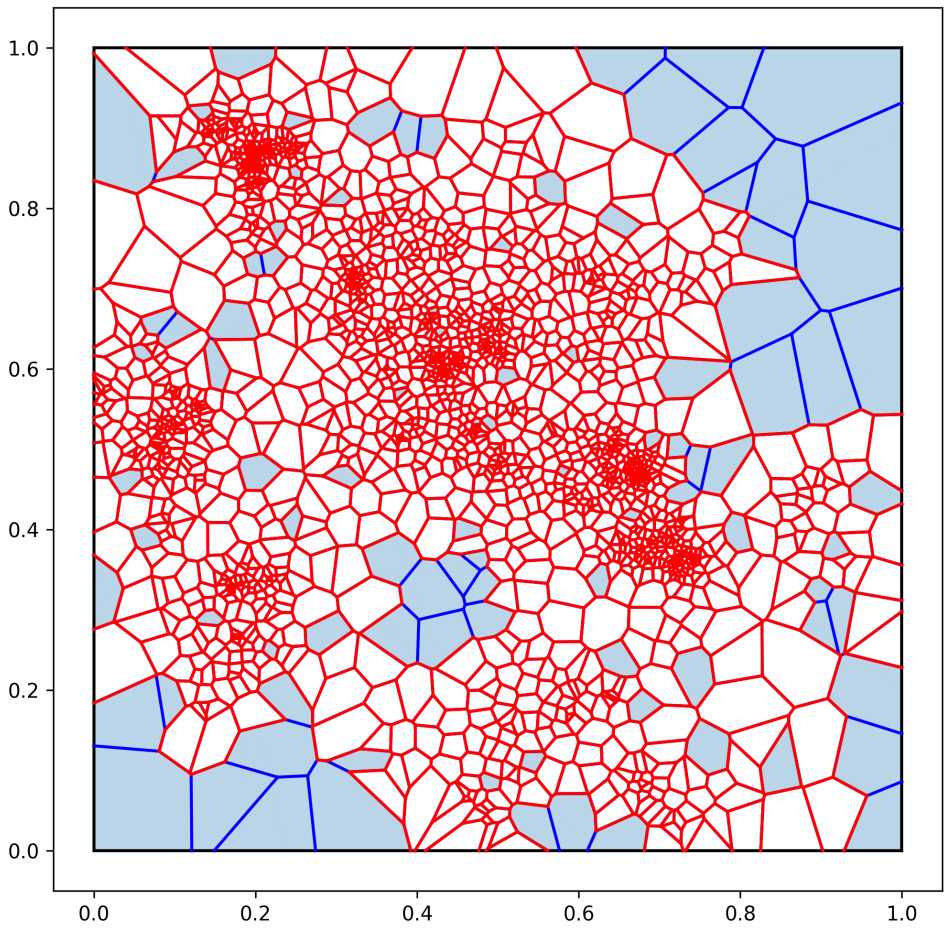
```
n = 100
id_ = 1
n_spec_list = 14 # 5%

fig, ax = plt.subplots(1, 1)
for i, coeff in zip([1,2,3], [2,3,3]):
    results = pd.read_csv(f'results/n{n}-id{id_}-{k}-{i}-{coeff*n}-10/result

    ax.plot(results['frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs frac n{n}-id{id_}-spec-{k}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()

fig, ax = plt.subplots(1, 1)
for i, coeff in zip([1,2,3], [2,3,3]):
    results = pd.read_csv(f'results/n{n}-id{id_}-{k}-{i}-{coeff*n}-10/result
```

```
    ax.plot(results['vol_frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs vol_frac n{n}-id{id_}-spec-{k}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()
```

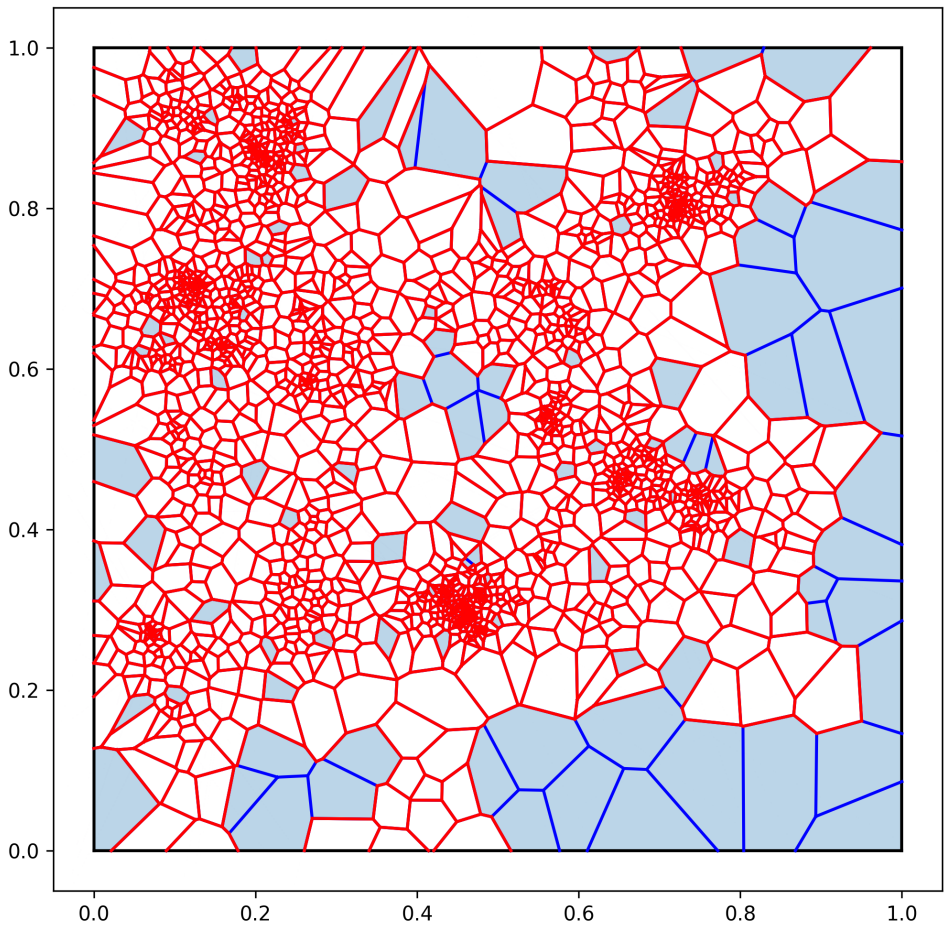




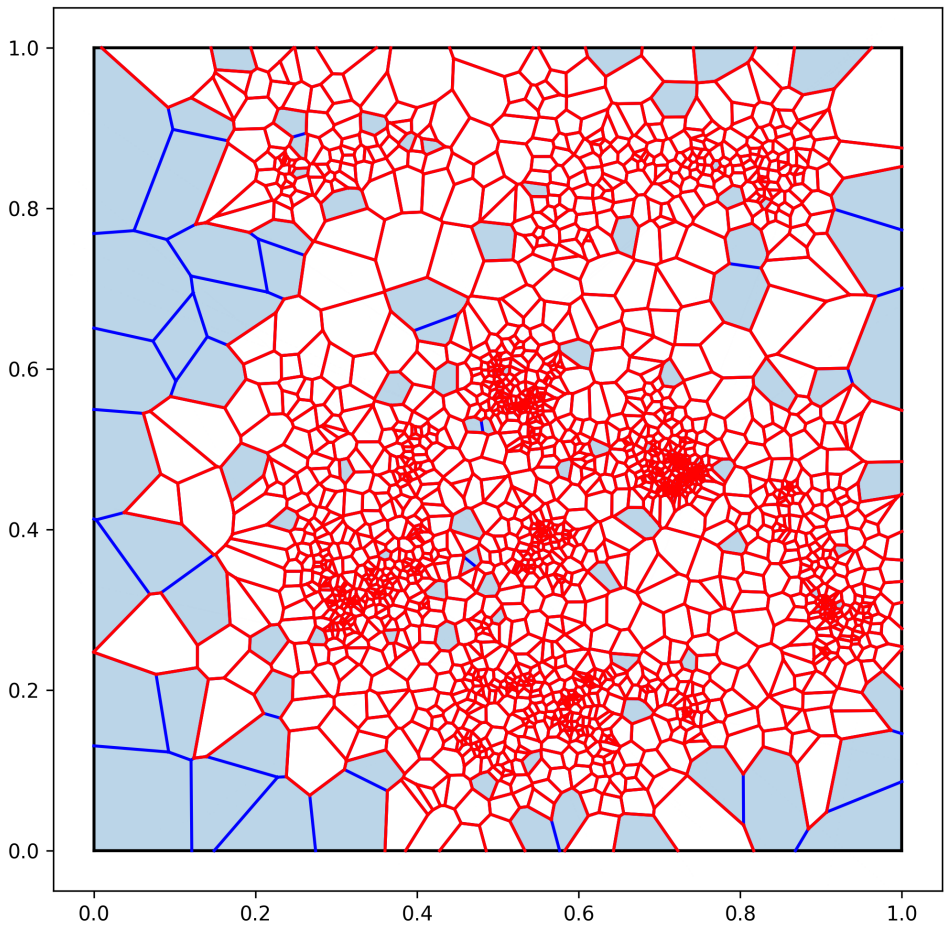

n = 100, id = 1, spec = 14-1, N_new_grains = 2000 (20 steps with 100 new)




n = 100, id = 1, spec = 14-2, N_new_grains = 2000 (20 steps with 100 new)



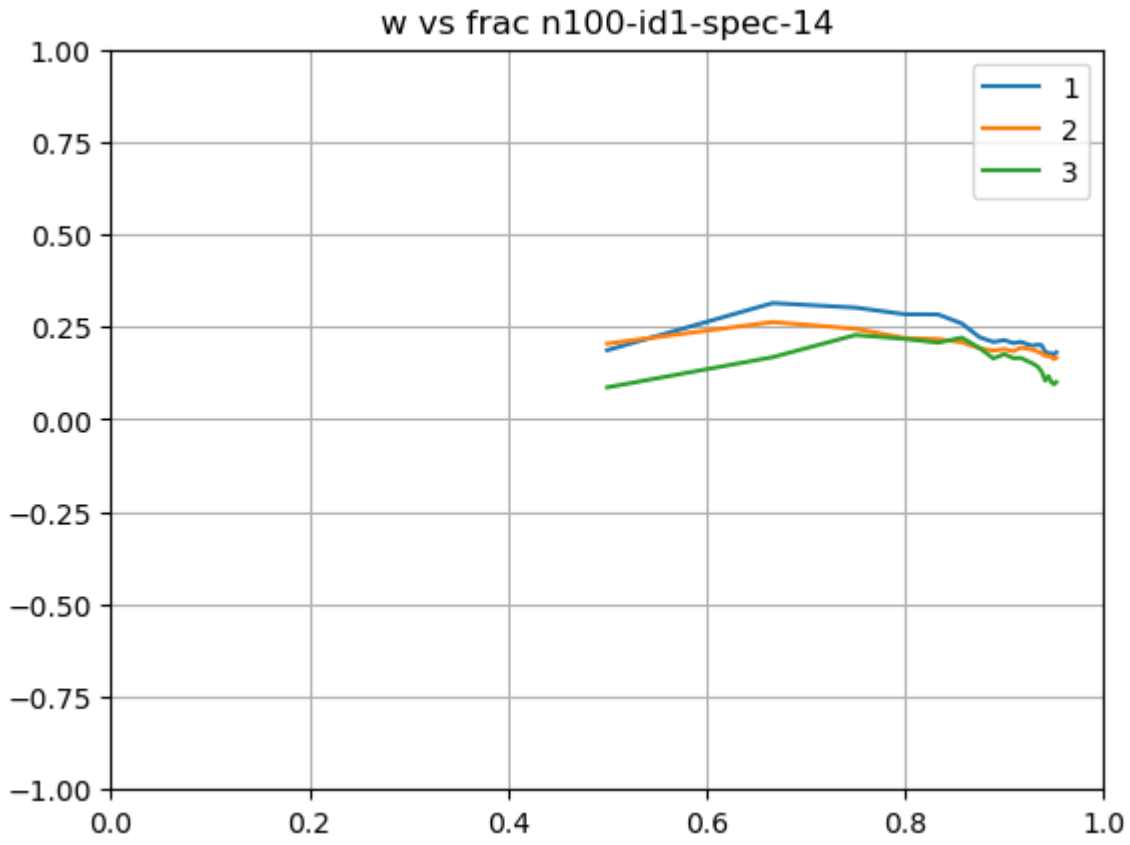

n = 100, id = 1, spec = 14-3, N_new_grains = 2000 (20 steps with 100 new)

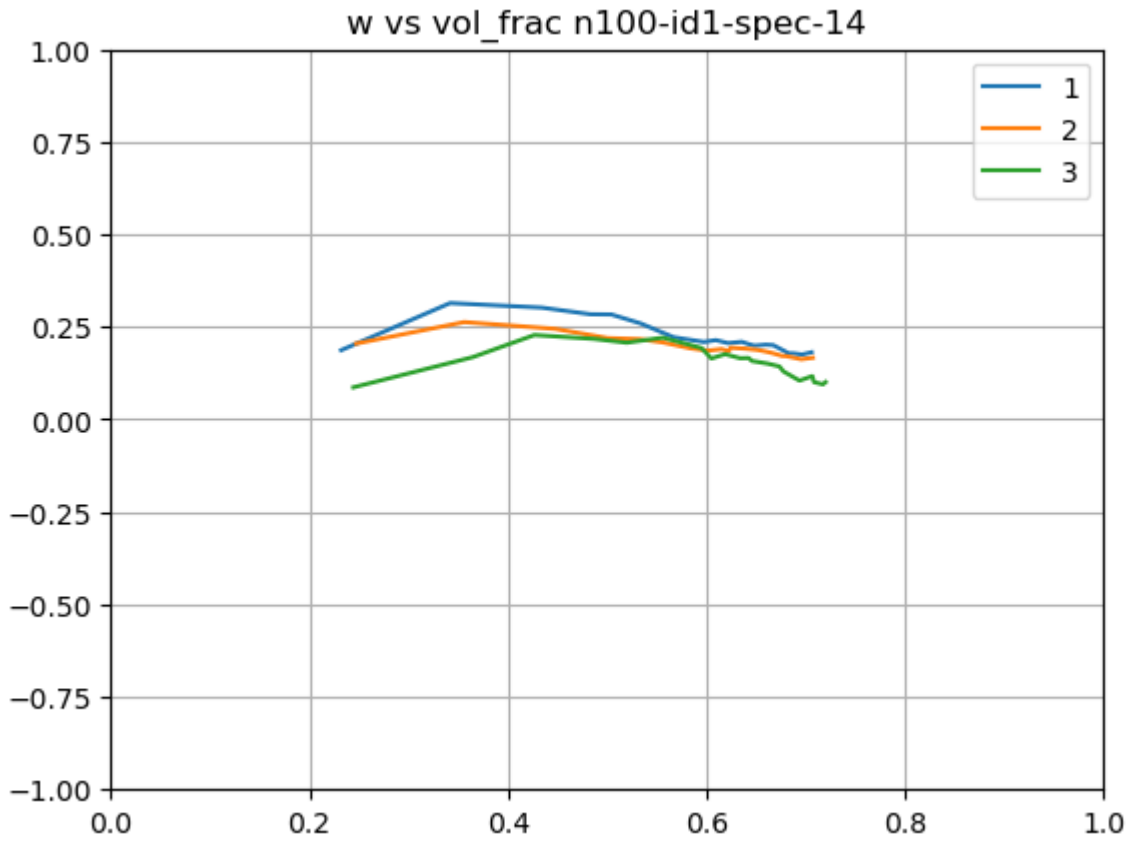



In [103…

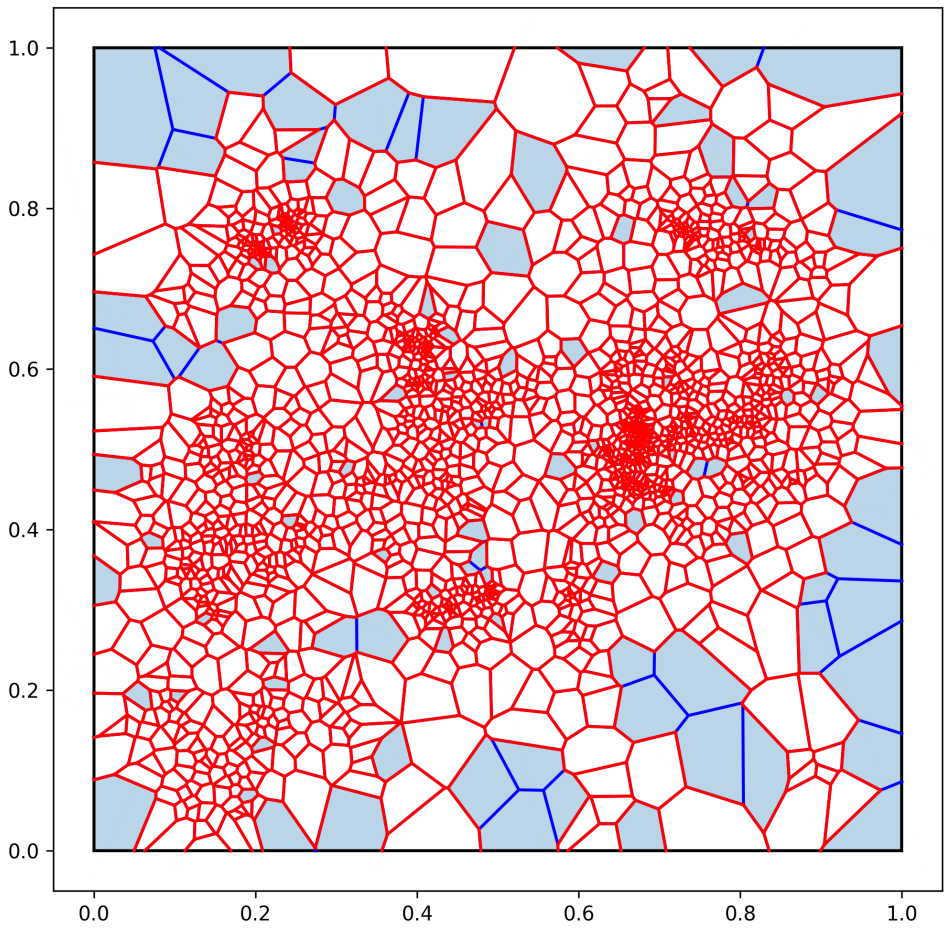
```
n = 100
id_ = 1
n_spec_list = 14 # 5%

fig, ax = plt.subplots(1, 1)
for i in [1,2,3]:
    results = pd.read_csv(f'results/n{n}-id{id_}-{k}-{i}-20-100/results.txt'

    ax.plot(results['frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs frac n{n}-id{id_}-spec-{k}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()

fig, ax = plt.subplots(1, 1)
for i in [1,2,3]:
    results = pd.read_csv(f'results/n{n}-id{id_}-{k}-{i}-20-100/results.txt'
```

```
    ax.plot(results['vol_frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs vol_frac n{n}-id{id_}-spec-{k}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()
```

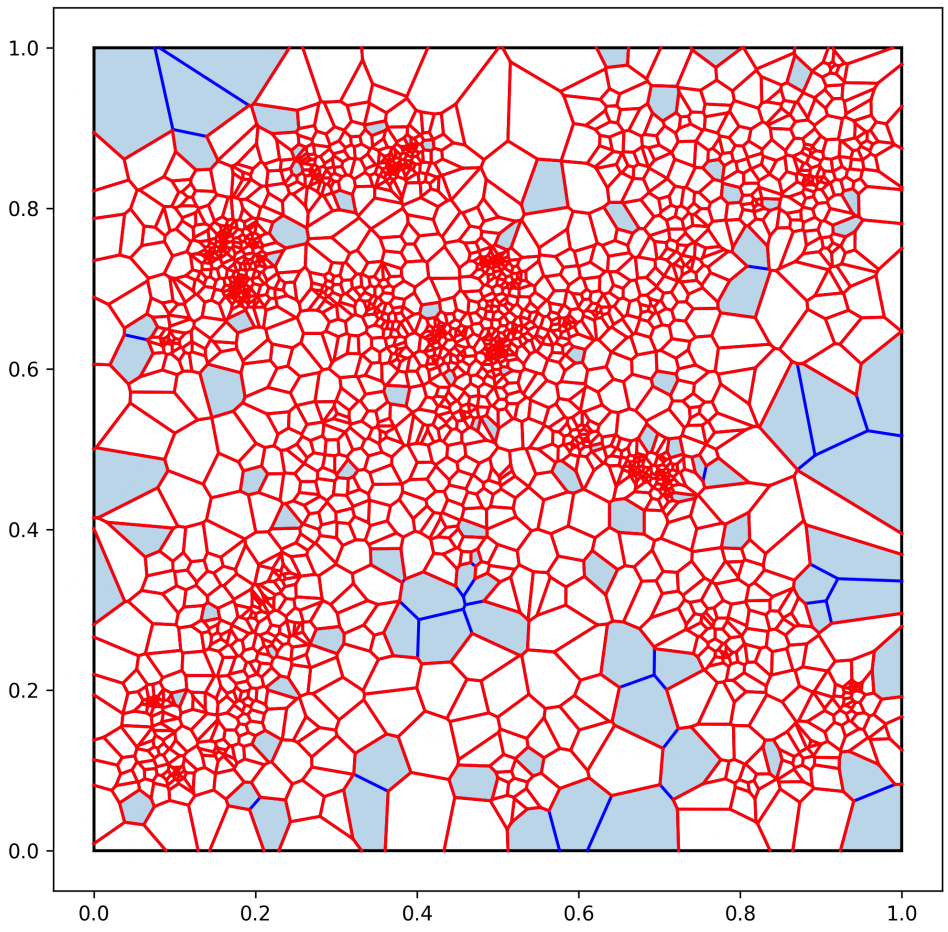




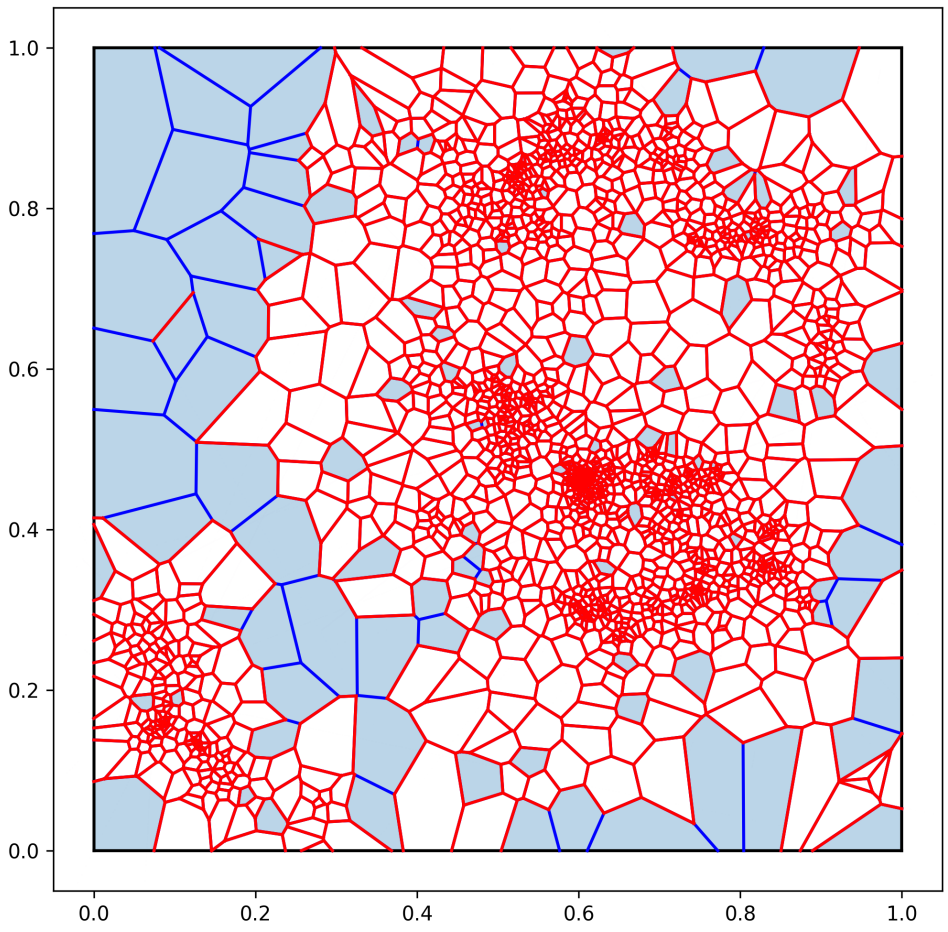

n = 100, id = 1, spec = 27-1, N_new_grains = 2000 (200 steps with 10 new)




n = 100, id = 1, spec = 27-2, N_new_grains = 2000 (200 steps with 10 new)



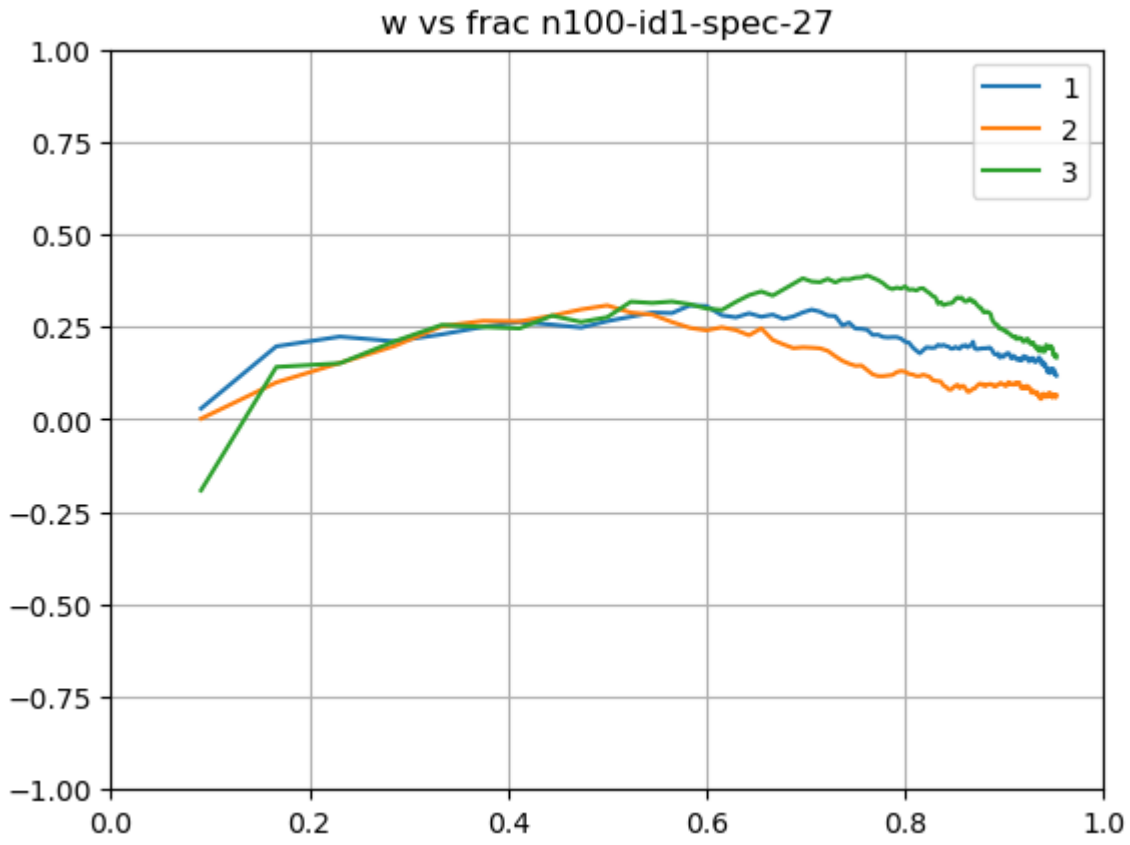

n = 100, id = 1, spec = 27-3, N_new_grains = 2000 (200 steps with 10 new)

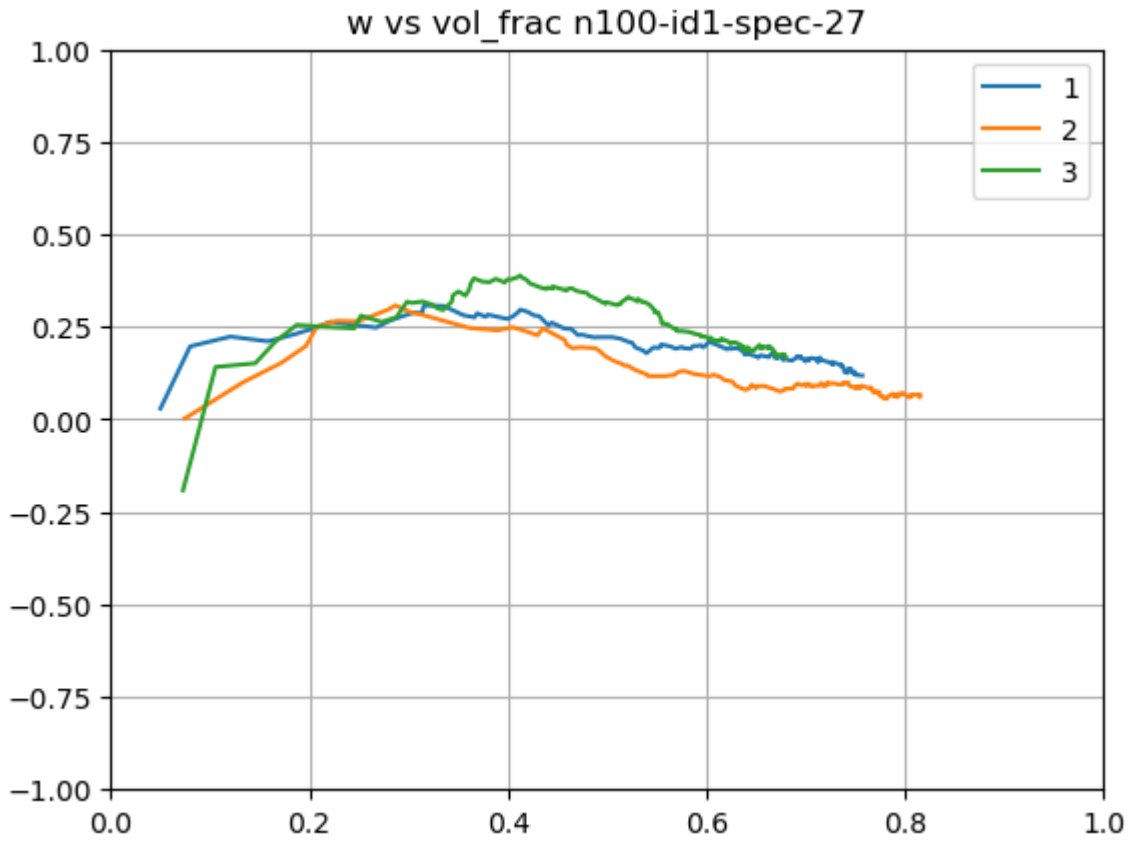



In [111…

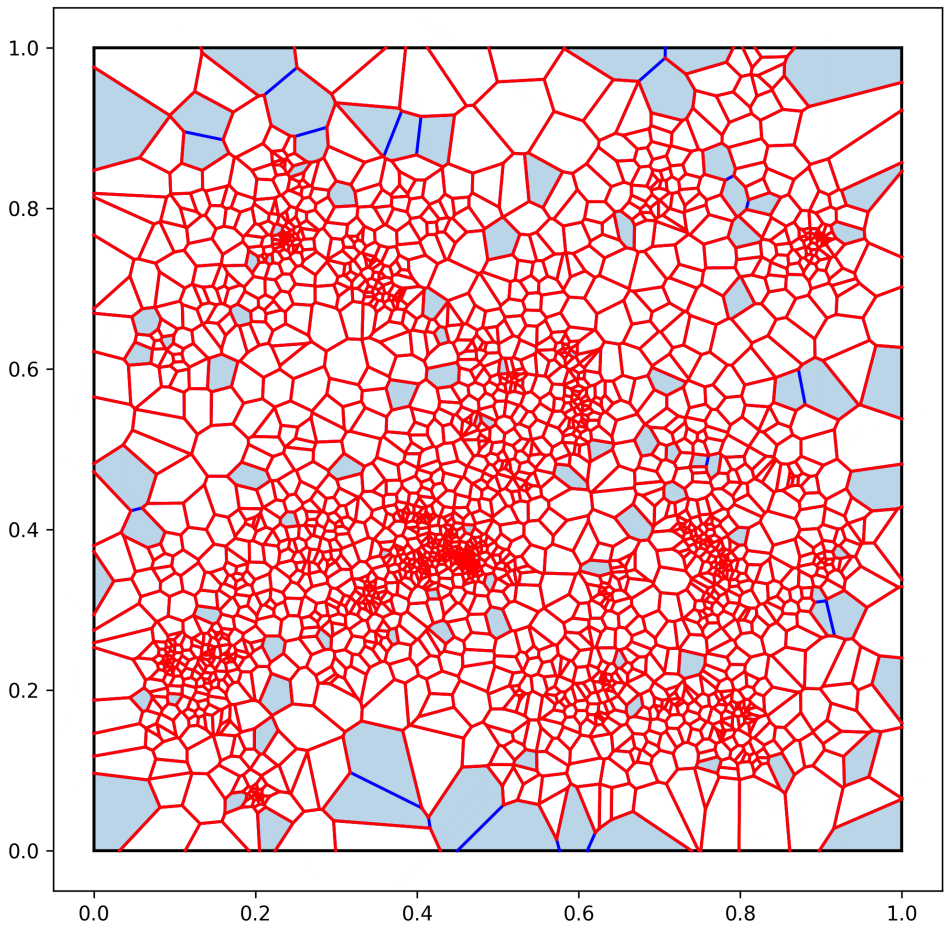
```
n = 100
id_ = 1
n_spec = 27 # 10%

fig, ax = plt.subplots(1, 1)
for i in [1,2,3]:
    results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{2*n}-10/resul

    ax.plot(results['frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs frac n{n}-id{id_}-spec-{n_spec}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()

fig, ax = plt.subplots(1, 1)
for i in [1,2,3]:
    results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{2*n}-10/resul
```

```
    ax.plot(results['vol_frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs vol_frac n{n}-id{id_}-spec-{n_spec}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()
```

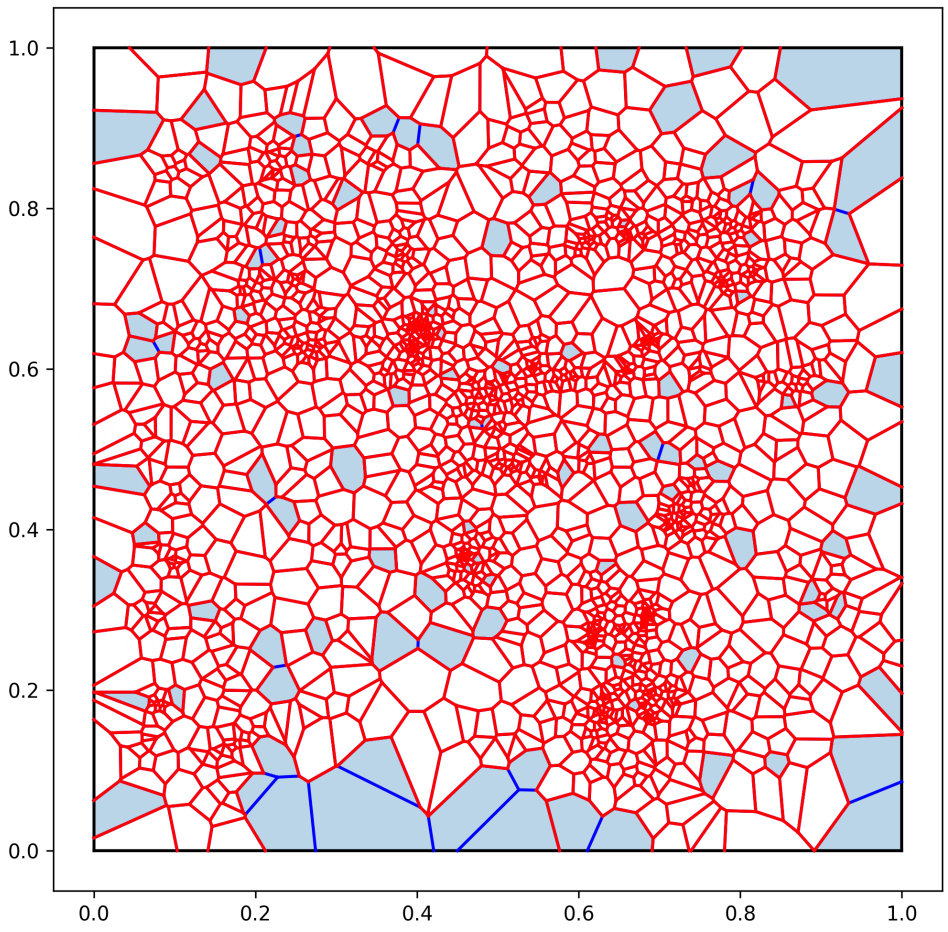




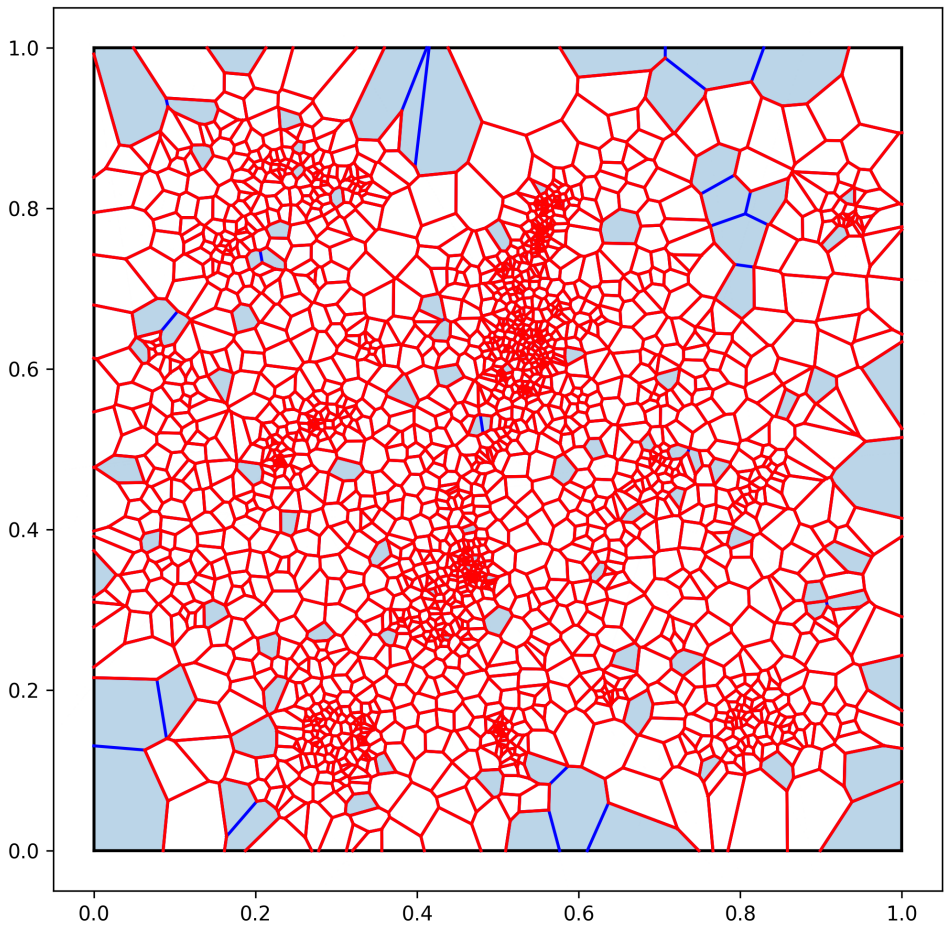

n = 100, id = 1, spec = 81-1, N_new_grains = 2000 (200 steps with 10 new)




n = 100, id = 1, spec = 81-2, N_new_grains = 2000 (200 steps with 10 new)



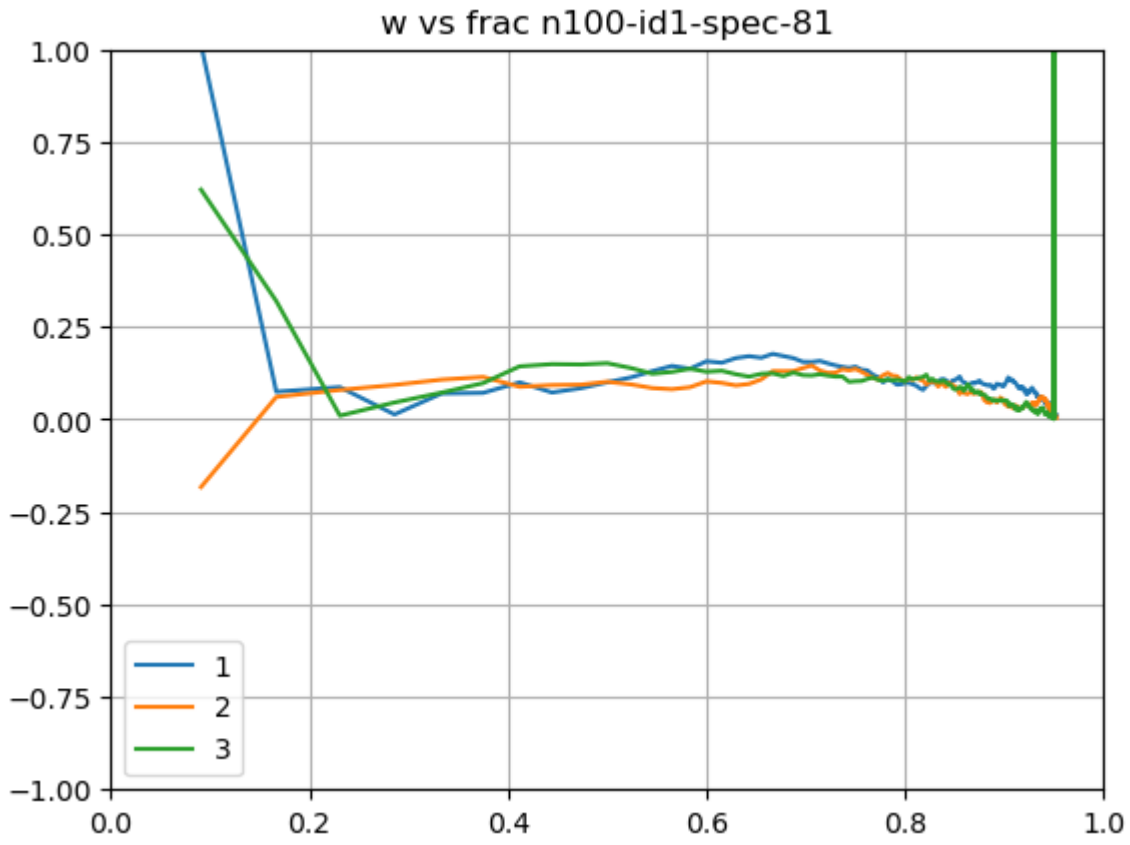

n = 100, id = 1, spec = 81-3, N_new_grains = 2000 (200 steps with 10 new)

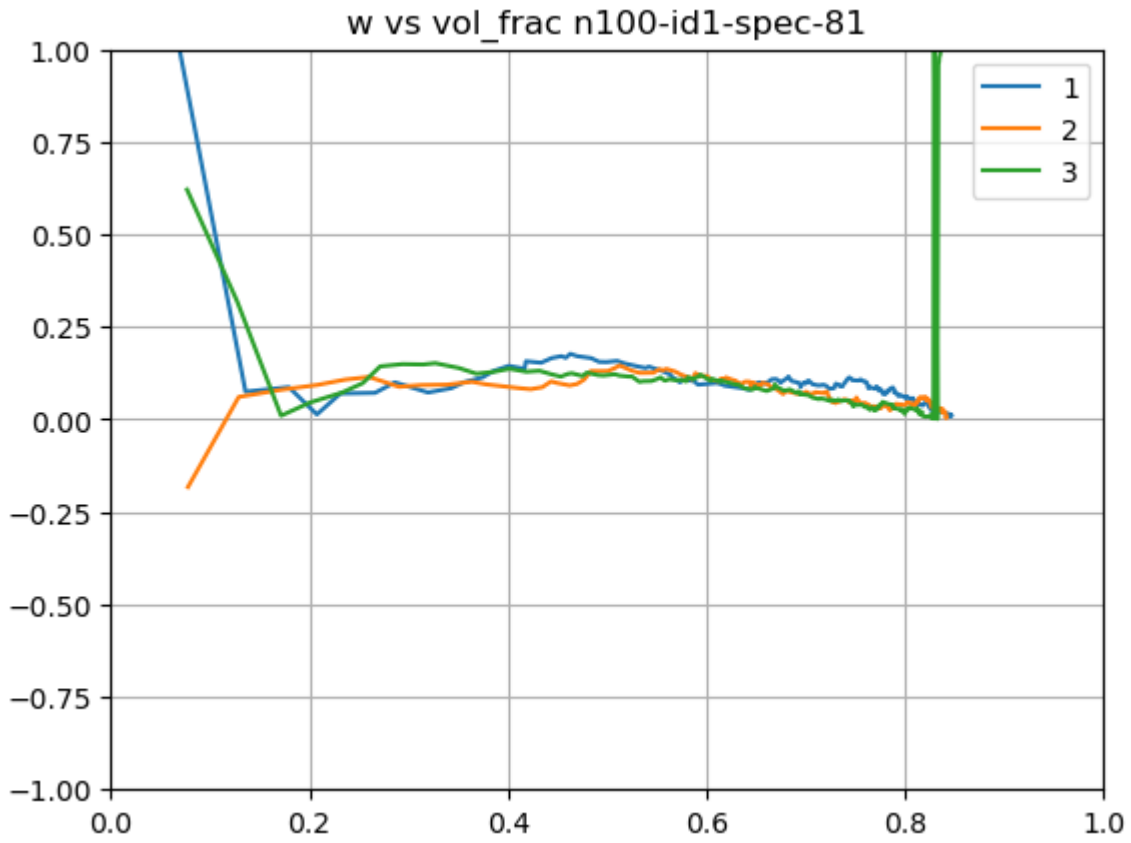



In [119…

```
n = 100
id_ = 1
n_spec = 81 # 10%

fig, ax = plt.subplots(1, 1)
for i in [1,2,3]:
    results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{2*n}-10/resul

    ax.plot(results['frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs frac n{n}-id{id_}-spec-{n_spec}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()

fig, ax = plt.subplots(1, 1)
for i in [1,2,3]:
    results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{2*n}-10/resul
```

```
    ax.plot(results['vol_frac'], results['omega'], label=str(i))
    ax.set_title(f'w vs vol_frac n{n}-id{id_}-spec-{n_spec}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])
plt.grid()
plt.show()
plt.close()
```

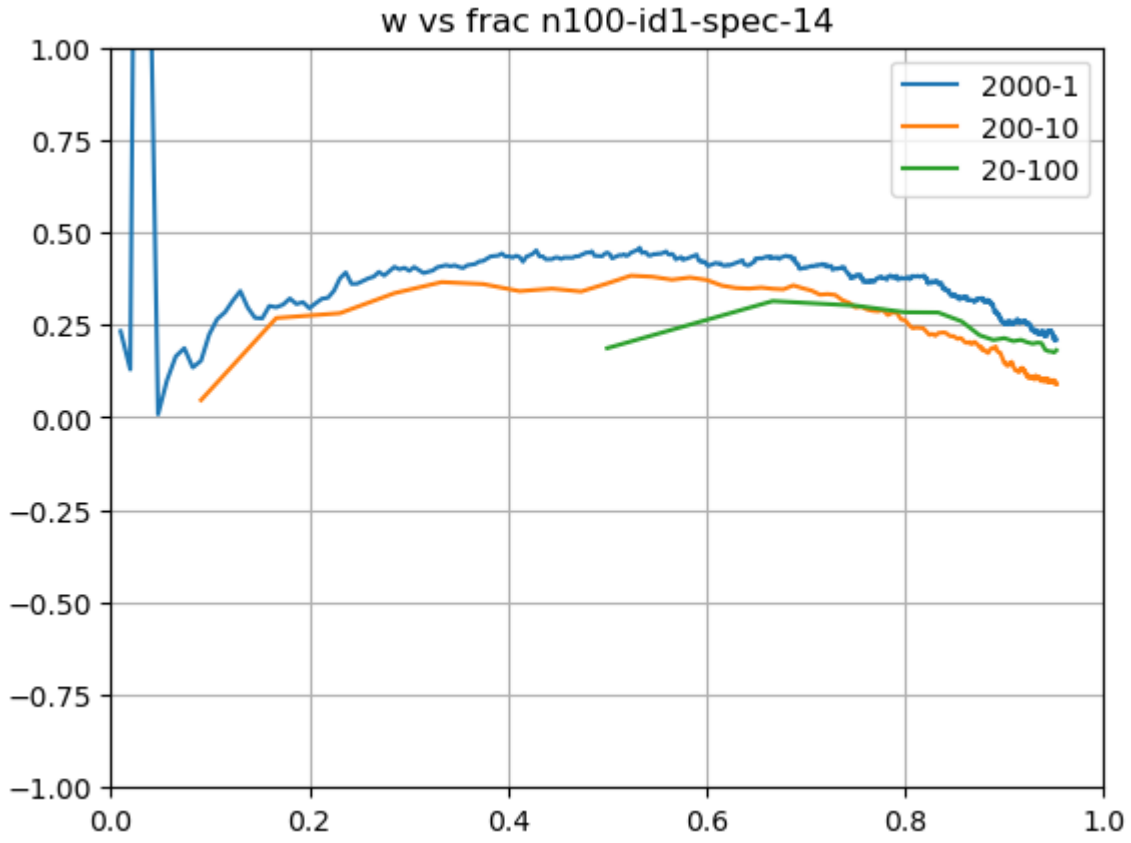





In [104…

```
n = 100
id_ = 1
n_spec = 14 # 5%

i = 1

fig, ax = plt.subplots(1, 1)

results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{20*n}-1/results.t
ax.plot(results['frac'], results['omega'], label='2000-1')

results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{2*n}-10/results.t
ax.plot(results['frac'], results['omega'], label='200-10')

results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-20-100/results.txt
ax.plot(results['frac'], results['omega'], label='20-100')

ax.set_title(f'w vs frac n{n}-id{id_}-spec-{k}')
ax.legend()
ax.set_ylim([-1, 1])
ax.set_xlim([0, 1])

plt.grid()
plt.show()
plt.close()

# fig, ax = plt.subplots(1, 1)
# for i in [1,2,3]:
#     results = pd.read_csv(f'results/n{n}-id{id_}-{k}-{i}-{2*n}-10/results.
```

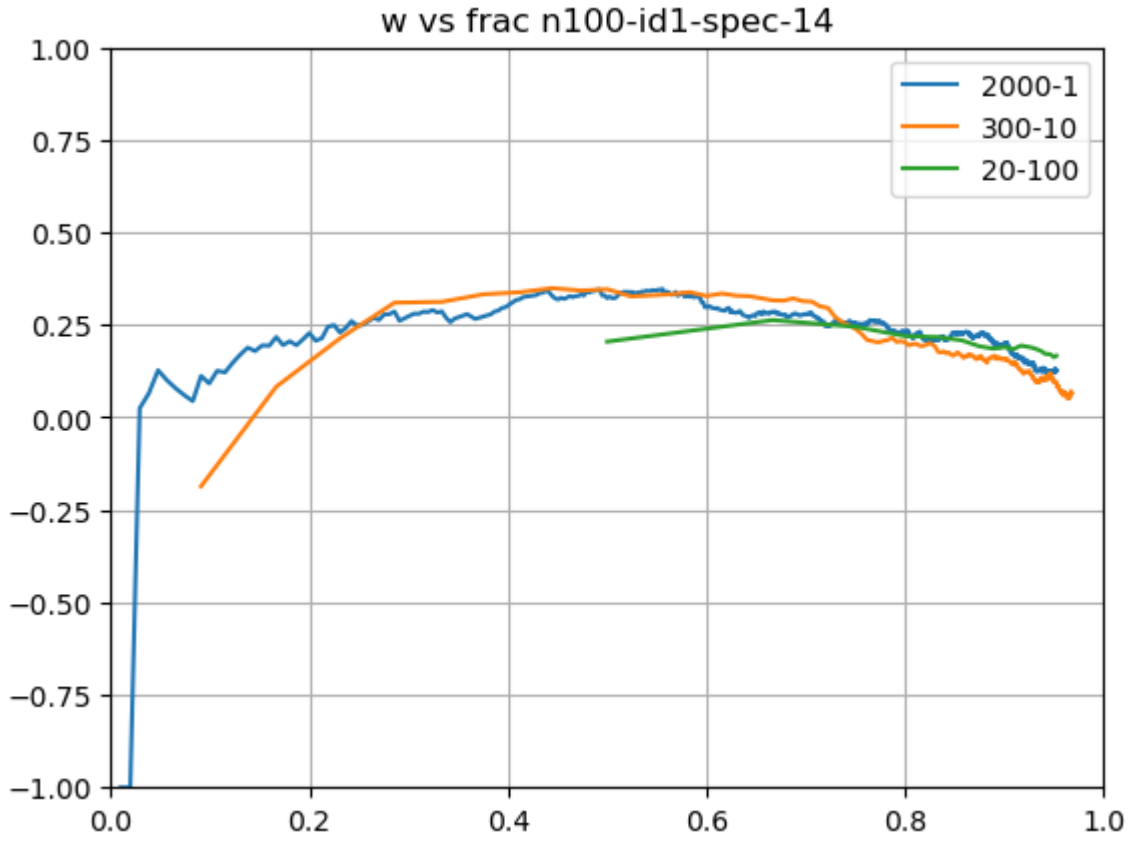
```
#     ax.plot(results['vol_frac'], results['omega'], label=str(i))
#     ax.set_title(f'w vs vol_frac n{n}-id{id_}-spec-{k}')
#     ax.legend()
#     ax.set_ylim([-1, 1])
#     ax.set_xlim([0, 1])
# plt.grid()
# plt.show()
# plt.close()
```




In [107…

```
n = 100
id_ = 1
n_spec = 14 # 5%

i = 2

fig, ax = plt.subplots(1, 1)

results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{20*n}-1/results.t
ax.plot(results['frac'], results['omega'], label='2000-1')

results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{3*n}-10/results.t
ax.plot(results['frac'], results['omega'], label='300-10')

results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-20-100/results.txt
ax.plot(results['frac'], results['omega'], label='20-100')

ax.set_title(f'w vs frac n{n}-id{id_}-spec-{k}')
ax.legend()
ax.set_ylim([-1, 1])
```

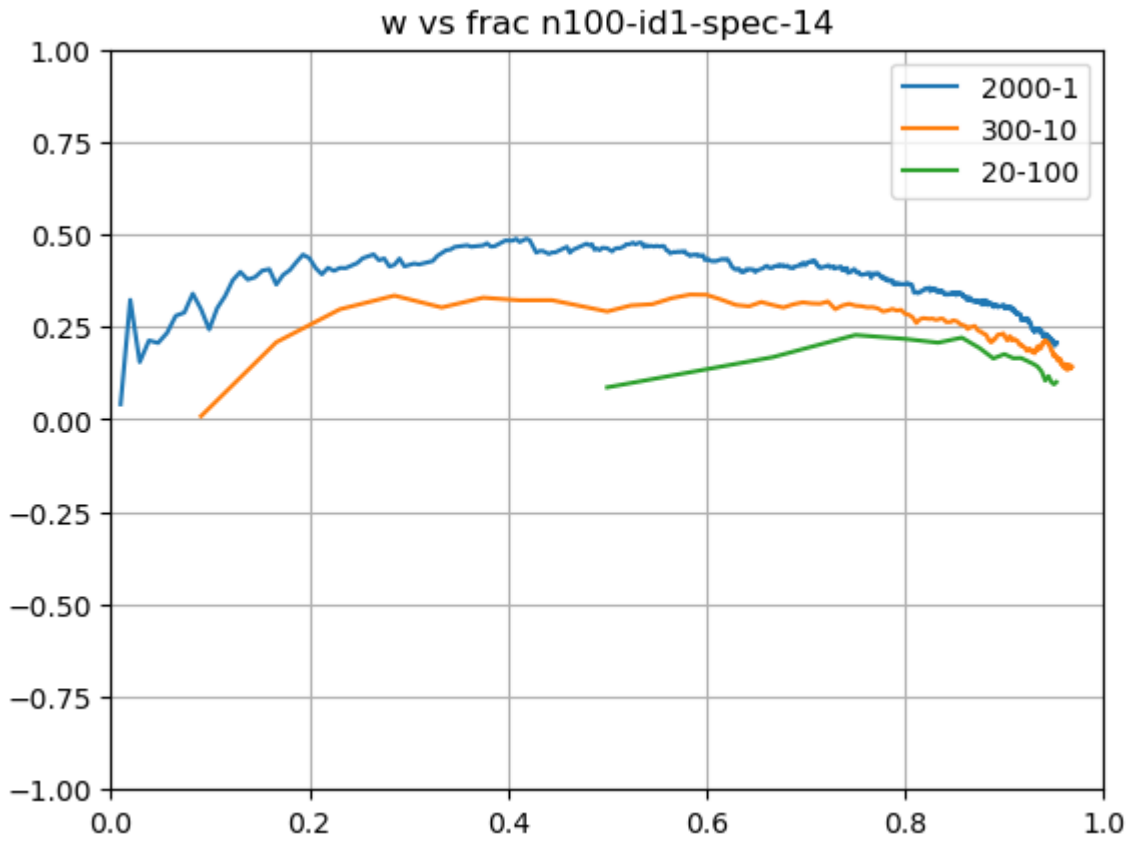
```
ax.set_xlim([0, 1])

plt.grid()
plt.show()
plt.close()

# fig, ax = plt.subplots(1, 1)
# for i in [1,2,3]:
#     results = pd.read_csv(f'results/n{n}-id{id_}-{k}-{i}-{2*n}-10/results.

#     ax.plot(results['vol_frac'], results['omega'], label=str(i))
#     ax.set_title(f'w vs vol_frac n{n}-id{id_}-spec-{k}')
#     ax.legend()
#     ax.set_ylim([-1, 1])
#     ax.set_xlim([0, 1])
# plt.grid()
# plt.show()
# plt.close()
```




In [108…

```
n = 100
id_ = 1
n_spec = 14 # 5%

i = 3

fig, ax = plt.subplots(1, 1)

results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{20*n}-1/results.t
ax.plot(results['frac'], results['omega'], label='2000-1')
```

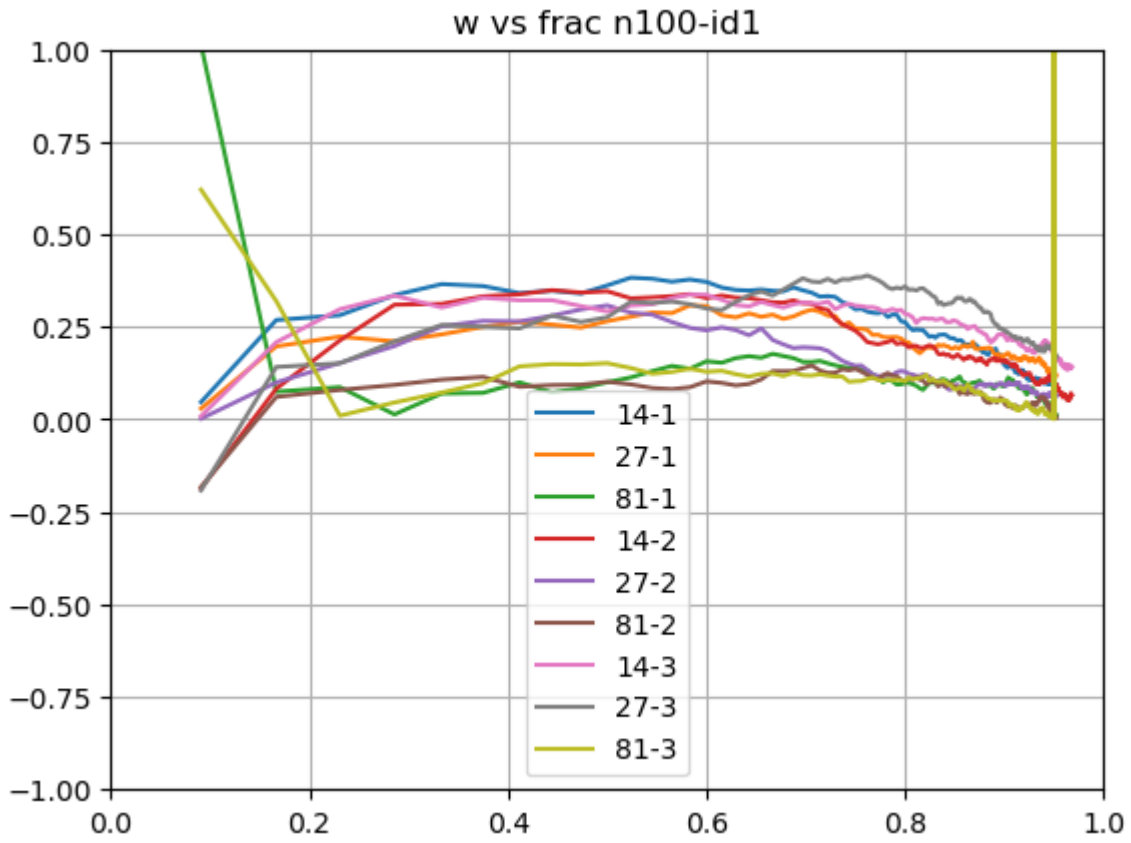
```
results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-{3*n}-10/results.t
ax.plot(results['frac'], results['omega'], label='300-10')

results = pd.read_csv(f'results/n{n}-id{id_}-{n_spec}-{i}-20-100/results.txt
ax.plot(results['frac'], results['omega'], label='20-100')

ax.set_title(f'w vs frac n{n}-id{id_}-spec-{k}')
ax.legend()
ax.set_ylim([-1, 1])
ax.set_xlim([0, 1])

plt.grid()
plt.show()
plt.close()

# fig, ax = plt.subplots(1, 1)
# for i in [1,2,3]:
#     results = pd.read_csv(f'results/n{n}-id{id_}-{k}-{i}-{2*n}-10/results.

#     ax.plot(results['vol_frac'], results['omega'], label=str(i))
#     ax.set_title(f'w vs vol_frac n{n}-id{id_}-spec-{k}')
#     ax.legend()
#     ax.set_ylim([-1, 1])
#     ax.set_xlim([0, 1])
# plt.grid()
# plt.show()
# plt.close()
```




In [113…

```
n = 100
id_ = 1
```

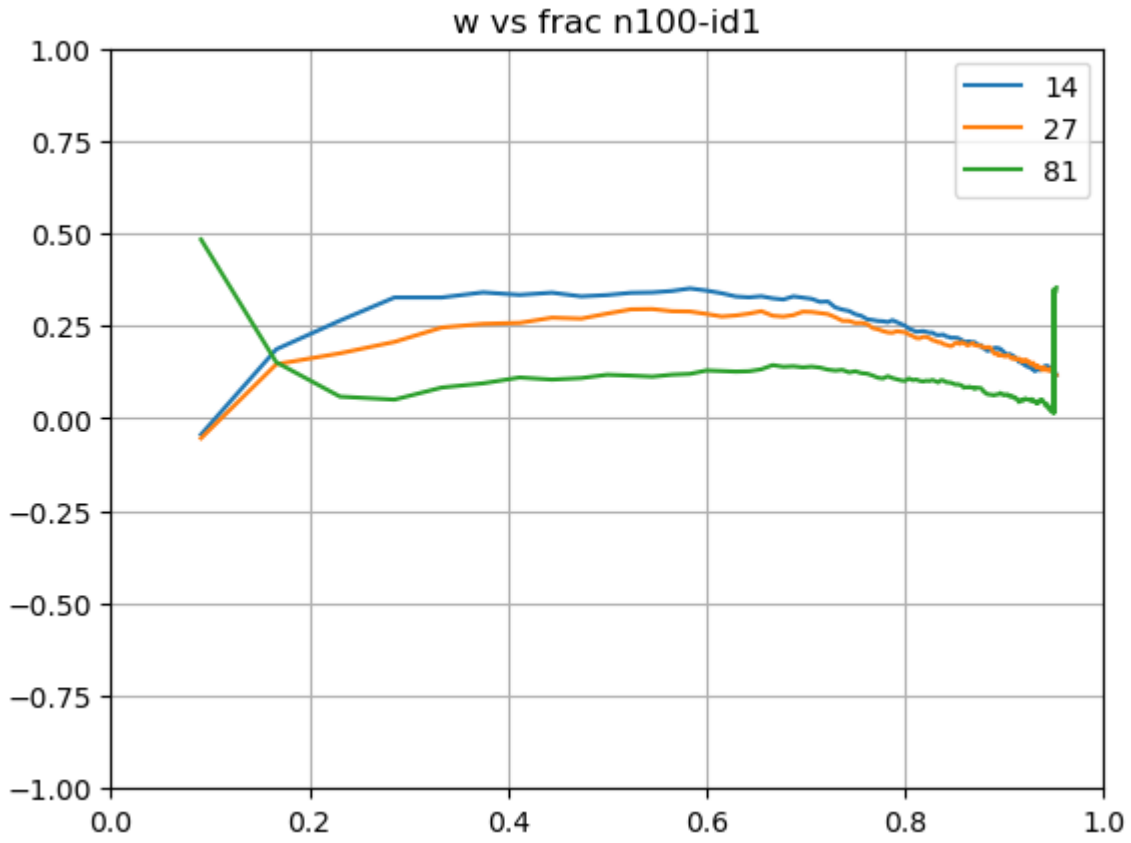
```
fig, ax = plt.subplots(1, 1)
for i in [1, 2, 3]:
    if i == 1:
        results = pd.read_csv(f'results/n{n}-id{id_}-14-{i}-200-10/results.t
    else:
        results = pd.read_csv(f'results/n{n}-id{id_}-14-{i}-300-10/results.t

    ax.plot(results['frac'], results['omega'], label=f'14-{i}')

    results = pd.read_csv(f'results/n{n}-id{id_}-27-{i}-200-10/results.txt')
    ax.plot(results['frac'], results['omega'], label=f'27-{i}')

    results = pd.read_csv(f'results/n{n}-id{id_}-81-{i}-200-10/results.txt')
    ax.plot(results['frac'], results['omega'], label=f'81-{i}')

    ax.set_title(f'w vs frac n{n}-id{id_}')
    ax.legend()
    ax.set_ylim([-1, 1])
    ax.set_xlim([0, 1])

plt.grid()
plt.show()
plt.close()

# fig, ax = plt.subplots(1, 1)
# for i in [1,2,3]:
#     results = pd.read_csv(f'results/n{n}-id{id_}-{k}-{i}-{2*n}-10/results.

#     ax.plot(results['vol_frac'], results['omega'], label=str(i))
#     ax.set_title(f'w vs vol_frac n{n}-id{id_}-spec-{k}')
#     ax.legend()
#     ax.set_ylim([-1, 1])
#     ax.set_xlim([0, 1])
# plt.grid()
# plt.show()
# plt.close()
```




In [118…

```
n = 100
id_ = 1

results = pd.read_csv(f'results/n{n}-id{id_}-14-1-200-10/results.txt')

# 14
results14_1 = pd.read_csv(f'results/n{n}-id{id_}-14-1-200-10/results.txt')['
results14_2 = pd.read_csv(f'results/n{n}-id{id_}-14-2-300-10/results.txt')['
results14_3 = pd.read_csv(f'results/n{n}-id{id_}-14-3-300-10/results.txt')['

results14 = (results14_1 + results14_2[:200] + results14_3[:200]) / 3

# 27
results27_1 = pd.read_csv(f'results/n{n}-id{id_}-27-1-200-10/results.txt')['
results27_2 = pd.read_csv(f'results/n{n}-id{id_}-27-2-200-10/results.txt')['
results27_3 = pd.read_csv(f'results/n{n}-id{id_}-27-3-200-10/results.txt')['

results27 = (results27_1 + results27_2 + results27_3) / 3

# 81
results81_1 = pd.read_csv(f'results/n{n}-id{id_}-81-1-200-10/results.txt')['
results81_2 = pd.read_csv(f'results/n{n}-id{id_}-81-2-200-10/results.txt')['
results81_3 = pd.read_csv(f'results/n{n}-id{id_}-81-3-200-10/results.txt')['

results81 = (results81_1 + results81_2 + results81_3) / 3

fig, ax = plt.subplots(1, 1)

ax.plot(results['frac'], results14, label='14')
```

```
ax.plot(results['frac'], results27, label='27')
ax.plot(results['frac'], results81, label='81')

ax.set_title(f'w vs frac n{n}-id{id_}')
ax.legend()
ax.set_ylim([-1, 1])
ax.set_xlim([0, 1])

plt.grid()
plt.show()
plt.close()
```




In [ ]:

# Weird omega

In [27]:
```
initial_complex = CellComplex.from_tess_file('sim_examples/n100-id1.tess')
```

In [28]:
```
initial_complex
```

Out[28]:
```
<class CellComplex> 2D
202 vertices
301 edges
100 faces
```

In [29]:
```
special_filename = f'sim_examples/n100-id1-spec-14-1.txt'
special_ids = np.loadtxt(special_filename)
```

In [30]:
```
initial_complex.reset_special(special_ids=special_ids, warn_external=False)
pairs_with_special_GB = []
# find pairs of grains with special GB
for special_id in initial_complex.get_special_ids():
    pair = set(initial_complex._GBs[special_id].incident_ids)
    pairs_with_special_GB.append(pair)
```

In [31]:
```
special_ids
```

Out[31]:
```
array([164., 154., 294., 229., 225.,  69.,  66., 265., 217., 300., 153.,
       245., 287., 235.])
```

In [32]:
```
pairs_with_special_GB
```

Out[32]:
```
[{14, 52},
 {15, 18},
 {35, 40},
 {35, 97},
 {38, 88},
 {54, 68},
 {58, 75},
 {59, 60},
 {61, 86},
 {65, 98},
 {70, 95},
 {81, 99},
 {87, 95},
 {95, 100}]
```

In [36]:
```
final_complex = CellComplex.from_tess_file('results/test-seeds/n103-id1.tess
```

In [37]:
```
special_ids = []
for cell in final_complex._GBs.values():
    if set(cell.incident_ids) in pairs_with_special_GB:
            special_ids.append(cell.id)
```

In [38]:
```
special_ids
```

Out[38]:
```
[66, 69, 153, 154, 164, 218, 230, 236, 247, 269, 292, 300]
```

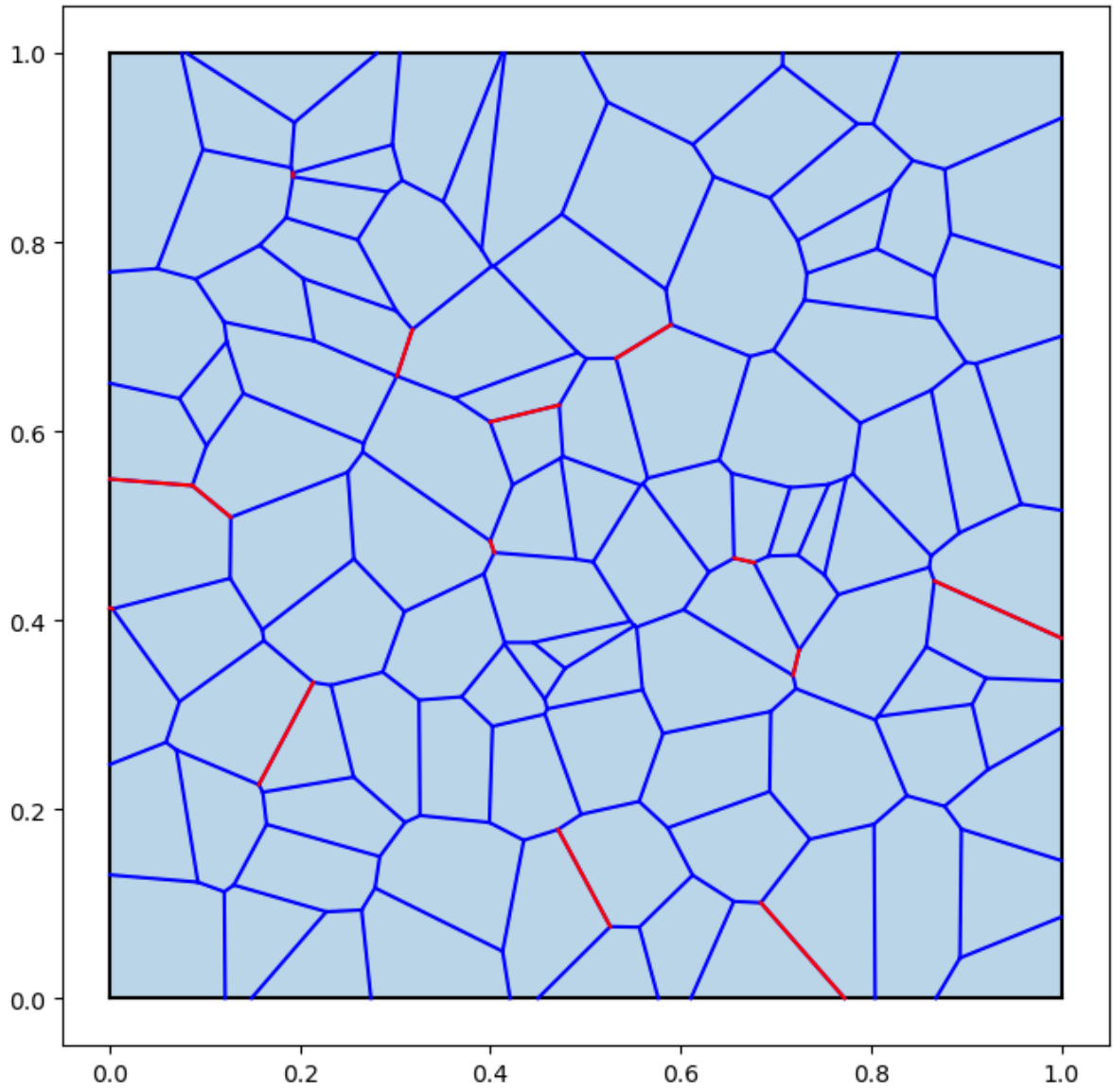
In [42]:
```
special_ids
```

Out[42]:
```
[66,
 69,
 153,
 154,
 164,
 218,
 230,
 236,
 247,
 269,
 292,
 300,
 226,
 239,
 276,
 310,
 60,
 298,
 268,
 306,
 309,
 253,
 227,
 297,
 307,
 275,
 310,
 120,
 190,
 255]
```

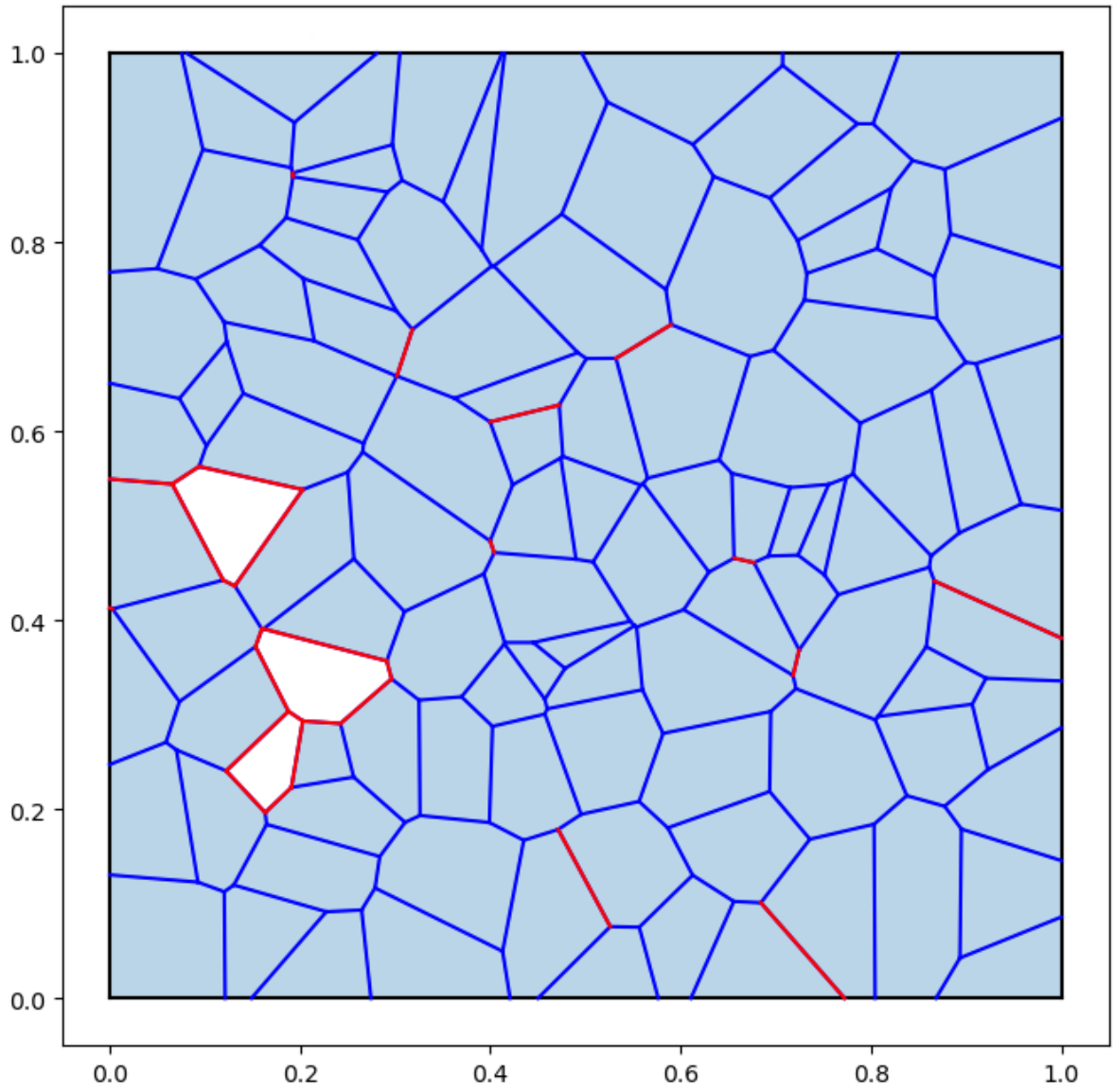
In [41]:
```
new_grains_total_size = 0
for grain_id in range(100 + 1, 103 + 1):
    gb_ids = final_complex._grains[grain_id].gb_ids
    special_ids += gb_ids
```

In [43]:
```
final_complex.reset_special(special_ids=set(special_ids), warn_external=Fals
```

In [46]:
```
cell_complex = initial_complex
ext_ids = cell_complex.get_external_ids('e')
int_ids = cell_complex.get_internal_ids('e')
spec_ids = cell_complex.get_special_ids()

ax = cell_complex.plot_edges(ext_ids, color='k')
cell_complex.plot_edges(int_ids, color='b', ax=ax)
cell_complex.plot_edges(spec_ids, color='r', ax=ax)
cell_complex.plot_faces(
    [*range(1, 100 + 1)], color='C0', alpha=0.3, ax=ax
)
plt.show()
```




In [48]:
```
cell_complex = final_complex
ext_ids = cell_complex.get_external_ids('e')
int_ids = cell_complex.get_internal_ids('e')
spec_ids = cell_complex.get_special_ids()

ax = cell_complex.plot_edges(ext_ids, color='k')
cell_complex.plot_edges(int_ids, color='b', ax=ax)
cell_complex.plot_edges(spec_ids, color='r', ax=ax)
cell_complex.plot_faces(
    [*range(1, 100 + 1)], color='C0', alpha=0.3, ax=ax
)
plt.show()
```




In [49]:

```
def omega(cell_complex: CellComplex, n0: int):
    """
    """
    D0 = 0
    D1 = 0
    D2 = 0
    w = 0
    number_of_internal_GBs = 0

    for gb in cell_complex._GBs.values():
        if not gb.is_external:
            number_of_internal_GBs += 1
            cell_ids = np.array(gb.incident_ids)
            d = (cell_ids > n0).sum()
            if d == 2:
                D2 += 1
            elif d == 1:
                D1 += 1
            elif d == 0:
                D0 += 1
```

```
            else:
                raise ValueError("Number of new grains must be 0, 1 or 2")

    D2 = D2 / number_of_internal_GBs
    D1 = D1 / number_of_internal_GBs
    D0 = D0 / number_of_internal_GBs

    n = cell_complex.grainnb # number of grains
    m = n - n0 # number of new grains
    p = m / n # fraction of new grains

    D0r = (1 - p) * (1 - p)
    D1r = 2 * (1 - p) * p
    D2r = p * p

    if p == 0 or p == 1:
        w = -9999
    elif D1 <= D1r:
        w = 1 - D1 / D1r
    elif D1 > D1r:
        w = D0 * D2 / D0r / D2r - 1

    return w
```

In [50]:
```
omega(final_complex, 100)
```

Out[50]:
```
3.310312538296569
```

In [ ]:
```

```

In [51]:
```
cell_complex = final_complex
n0 = 100

D0 = 0
D1 = 0
D2 = 0
w = 0
number_of_internal_GBs = 0

for gb in cell_complex._GBs.values():
    if not gb.is_external:
        number_of_internal_GBs += 1
        cell_ids = np.array(gb.incident_ids)
        d = (cell_ids > n0).sum()
        if d == 2:
            D2 += 1
        elif d == 1:
            D1 += 1
        elif d == 0:
            D0 += 1
        else:
            raise ValueError("Number of new grains must be 0, 1 or 2")

D2 = D2 / number_of_internal_GBs
D1 = D1 / number_of_internal_GBs
```

```
D0 = D0 / number_of_internal_GBs

n = cell_complex.grainnb # number of grains
m = n - n0 # number of new grains
p = m / n # fraction of new grains

D0r = (1 - p) * (1 - p)
D1r = 2 * (1 - p) * p
D2r = p * p

if p == 0 or p == 1:
    w = -9999
elif D1 <= D1r:
    w = 1 - D1 / D1r
elif D1 > D1r:
    w = D0 * D2 / D0r / D2r - 1
```

In [53]:
```
number_of_internal_GBs
```

Out[53]: 272

In [56]:
```
n = cell_complex.grainnb
```

In [57]:
```
n
```

Out[57]: 103

In [59]:
```
m = n - n0
m
```

Out[59]: 3

In [60]:
```
p = m / n
p
```

Out[60]: 0.02912621359223301

In [61]:
```
D0r = (1 - p) * (1 - p)
D0r
```

Out[61]: 0.9425959091337544

In [62]:
```
D1r = 2 * (1 - p) * p
D1r
```

Out[62]: 0.056555754548025264

In [63]:
```
D2r = p * p
D2r
```

Out[63]: 0.000848336318220379

In [66]:
```
w = 1 - D1 / D1r
```

In [67]:
```
w
```

Out[67]: -0.04009803921568622

In [75]:
```
D0
```

Out[75]: 0.9375

In [76]:
```
D1
```

Out[76]: 0.058823529411764705

In [77]:
```
D2
```

Out[77]: 0.003676470588235294

In [71]:
```
D0 * number_of_internal_GBs
```

Out[71]: 255.0

In [72]:
```
D1 * number_of_internal_GBs
```

Out[72]: 16.0

In [73]:
```
D2 * number_of_internal_GBs
```

Out[73]: 1.0

In [64]:
```
D0 + D1 + D2
```

Out[64]: 1.0

In [70]:
```
(D1 + 2*D2) / 2
```

Out[70]: 0.03308823529411765

In [74]:
```
(D1r + 2*D2r) / 2
```

Out[74]: 0.02912621359223301

In [65]:
```
D0r + D1r + D2r
```

Out[65]: 1.0

In [68]:
```
D0 * D2 / D0r / D2r - 1
```

Out[68]: 3.310312538296569

In [78]:
```
D1 / D1r
```

Out[78]: 1.0400980392156862

In [ ]:

In [2]:
```
for id_ in [1, 2, 42]:
    for n, ks in zip([10, 100, 1000], [[2, 6], [14, 27, 81], [145, 290, 870]
        for k in ks:
            for i in [1, 2, 3]:
                print(f'n{n}-id{id_}-{k}-{i}-{20*n}-1')
```

n10-id1-2-1-200-1
n10-id1-2-2-200-1
n10-id1-2-3-200-1
n10-id1-6-1-200-1
n10-id1-6-2-200-1
n10-id1-6-3-200-1
n100-id1-14-1-2000-1
n100-id1-14-2-2000-1
n100-id1-14-3-2000-1
n100-id1-27-1-2000-1
n100-id1-27-2-2000-1
n100-id1-27-3-2000-1
n100-id1-81-1-2000-1
n100-id1-81-2-2000-1
n100-id1-81-3-2000-1
n1000-id1-145-1-20000-1
n1000-id1-145-2-20000-1
n1000-id1-145-3-20000-1
n1000-id1-290-1-20000-1
n1000-id1-290-2-20000-1
n1000-id1-290-3-20000-1
n1000-id1-870-1-20000-1
n1000-id1-870-2-20000-1
n1000-id1-870-3-20000-1
n10-id2-2-1-200-1
n10-id2-2-2-200-1
n10-id2-2-3-200-1
n10-id2-6-1-200-1
n10-id2-6-2-200-1
n10-id2-6-3-200-1
n100-id2-14-1-2000-1
n100-id2-14-2-2000-1
n100-id2-14-3-2000-1
n100-id2-27-1-2000-1
n100-id2-27-2-2000-1
n100-id2-27-3-2000-1
n100-id2-81-1-2000-1
n100-id2-81-2-2000-1
n100-id2-81-3-2000-1
n1000-id2-145-1-20000-1
n1000-id2-145-2-20000-1
n1000-id2-145-3-20000-1
n1000-id2-290-1-20000-1
n1000-id2-290-2-20000-1
n1000-id2-290-3-20000-1
n1000-id2-870-1-20000-1
n1000-id2-870-2-20000-1
n1000-id2-870-3-20000-1
n10-id42-2-1-200-1
n10-id42-2-2-200-1
n10-id42-2-3-200-1
n10-id42-6-1-200-1
n10-id42-6-2-200-1
n10-id42-6-3-200-1
n100-id42-14-1-2000-1
n100-id42-14-2-2000-1

n100-id42-14-3-2000-1
n100-id42-27-1-2000-1
n100-id42-27-2-2000-1
n100-id42-27-3-2000-1
n100-id42-81-1-2000-1
n100-id42-81-2-2000-1
n100-id42-81-3-2000-1
n1000-id42-145-1-20000-1
n1000-id42-145-2-20000-1
n1000-id42-145-3-20000-1
n1000-id42-290-1-20000-1
n1000-id42-290-2-20000-1
n1000-id42-290-3-20000-1
n1000-id42-870-1-20000-1
n1000-id42-870-2-20000-1
n1000-id42-870-3-20000-1