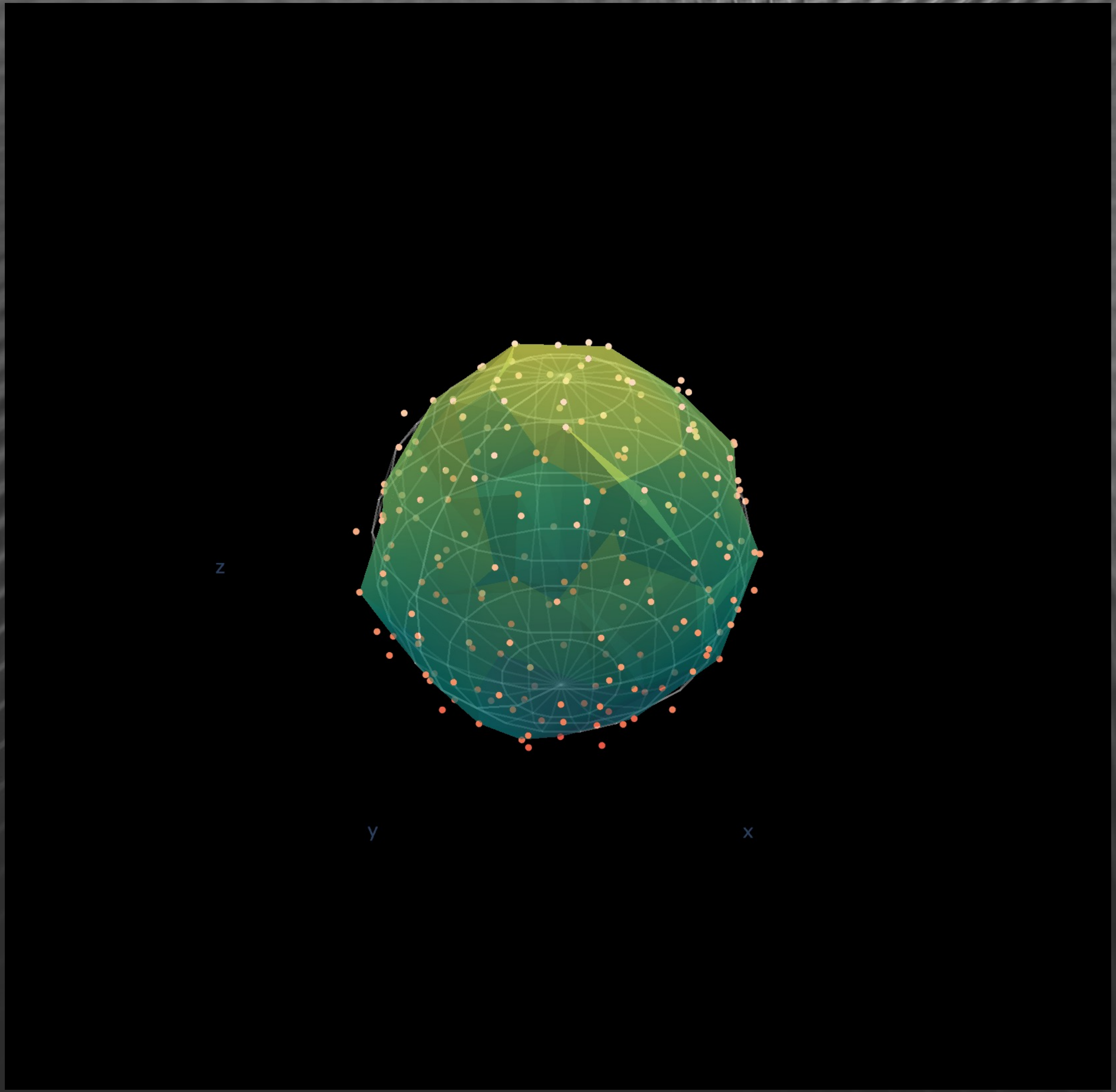
Open Applied Topology

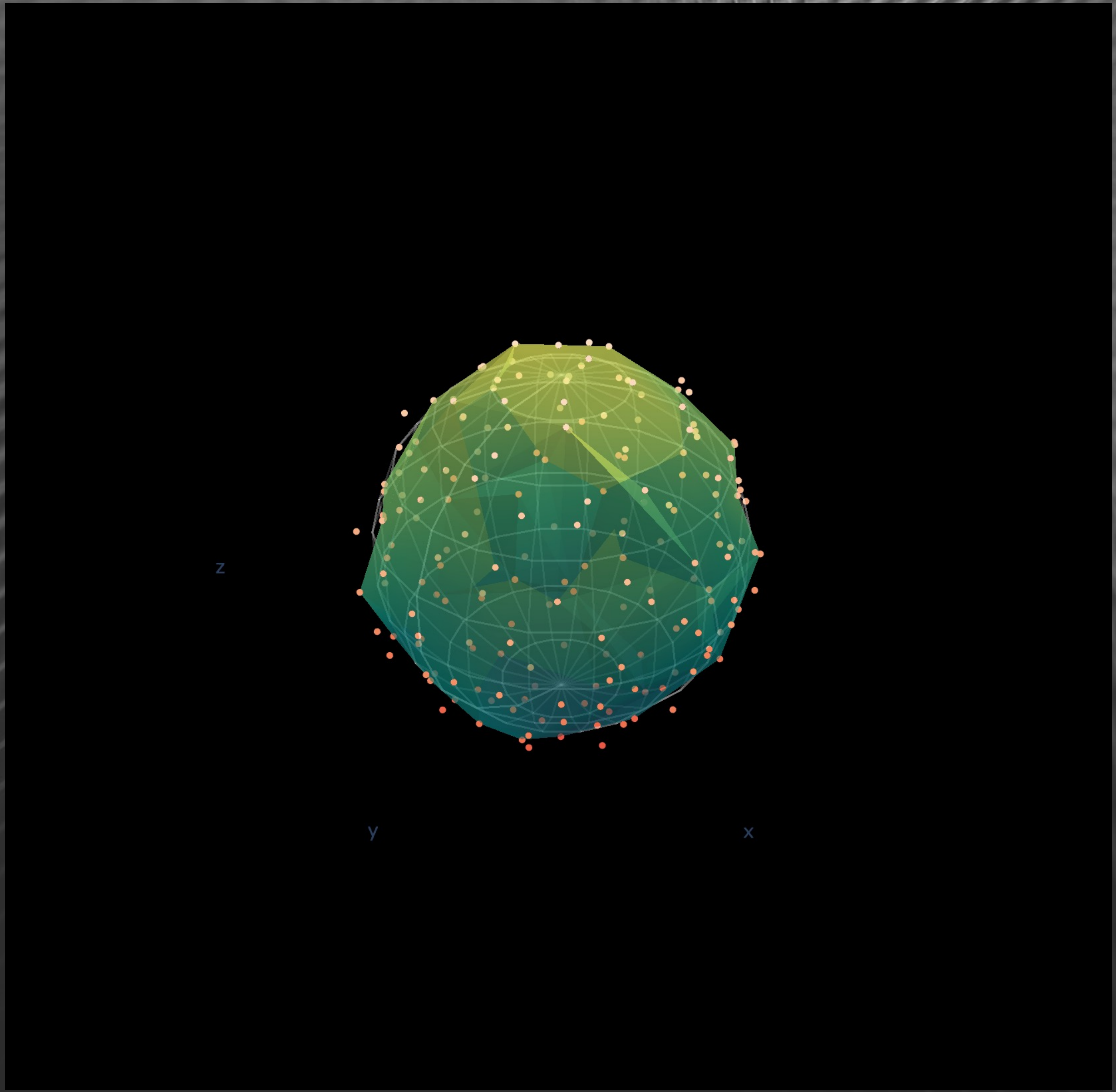
Gregory Henselman-Petrusek Roek
Pacific Northwest National Laboratory

PNNL is operated by Battelle for the U.S. Department of Energy



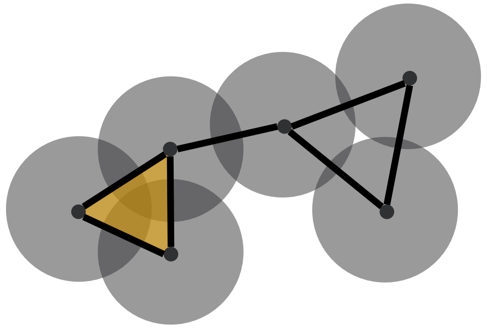
Pacific
Northwest
NATIONAL LABORATORY

Fast, user-friendly matrix algebra for applied topology



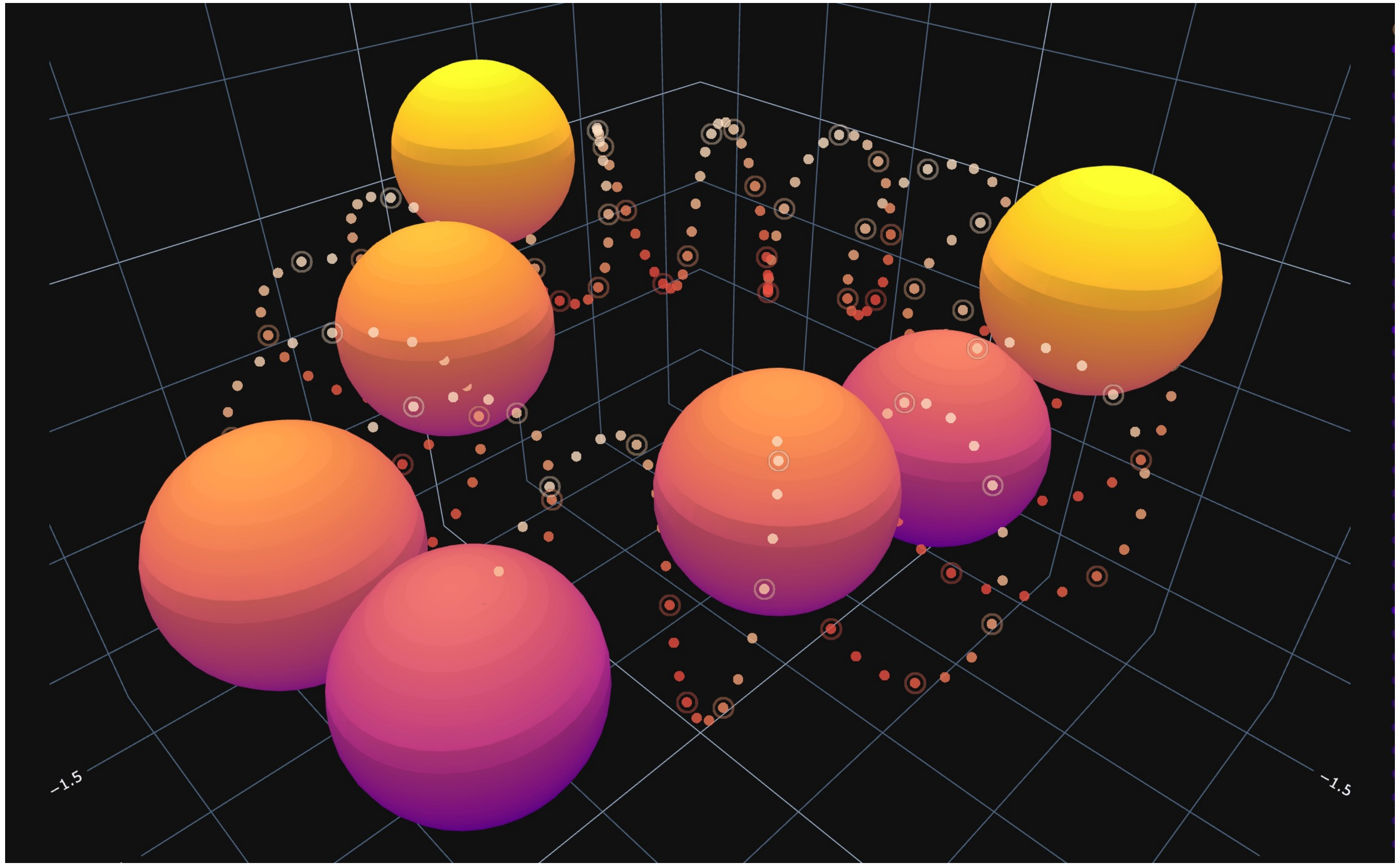
U.S. DEPARTMENT OF ENERGY
BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy



CW Complexes

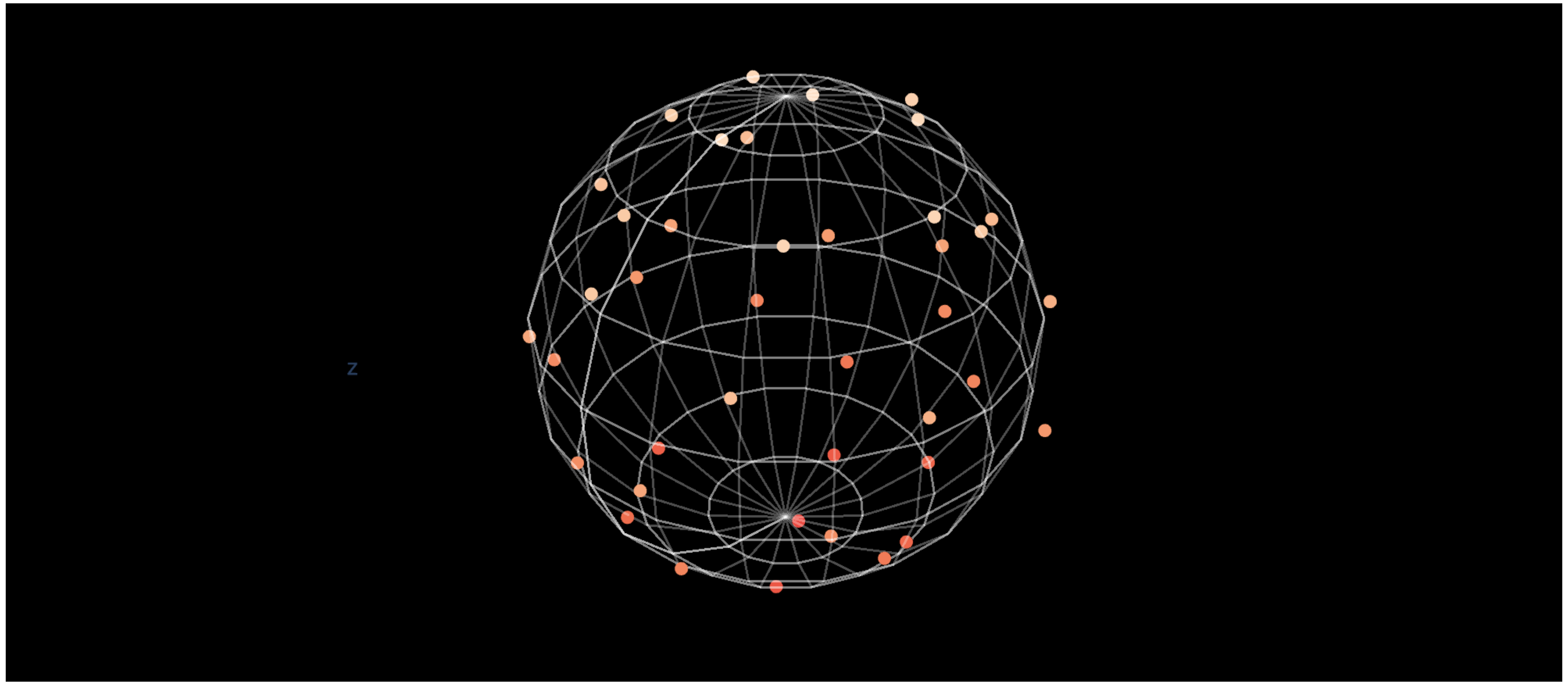
- Simplicial
  - Cech (point cloud)
  - Vietoris-Rips (point cloud)
  - Nerve (poset)
  - Dowker (toplexes)
- Cubical
  - (digital images)

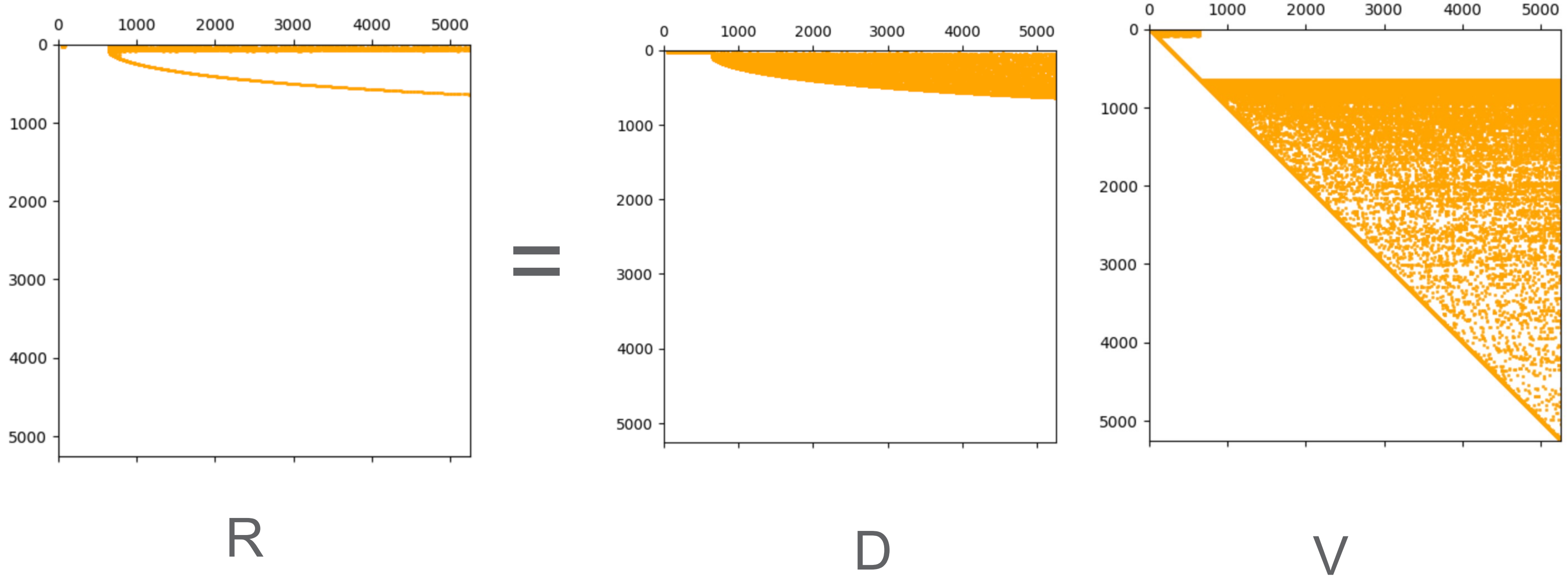


Sparse algebra

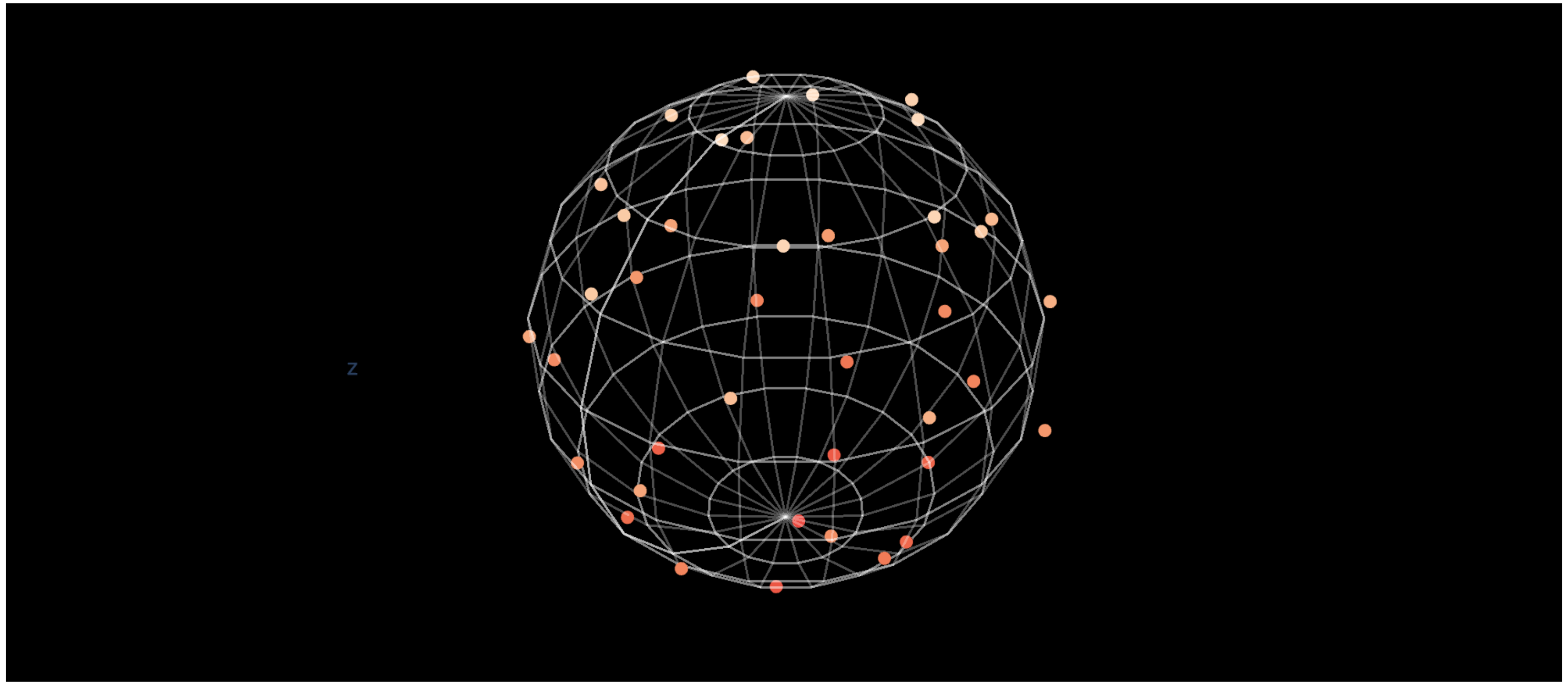
Matrices

- Multiplication
- Inversion
- Factorization



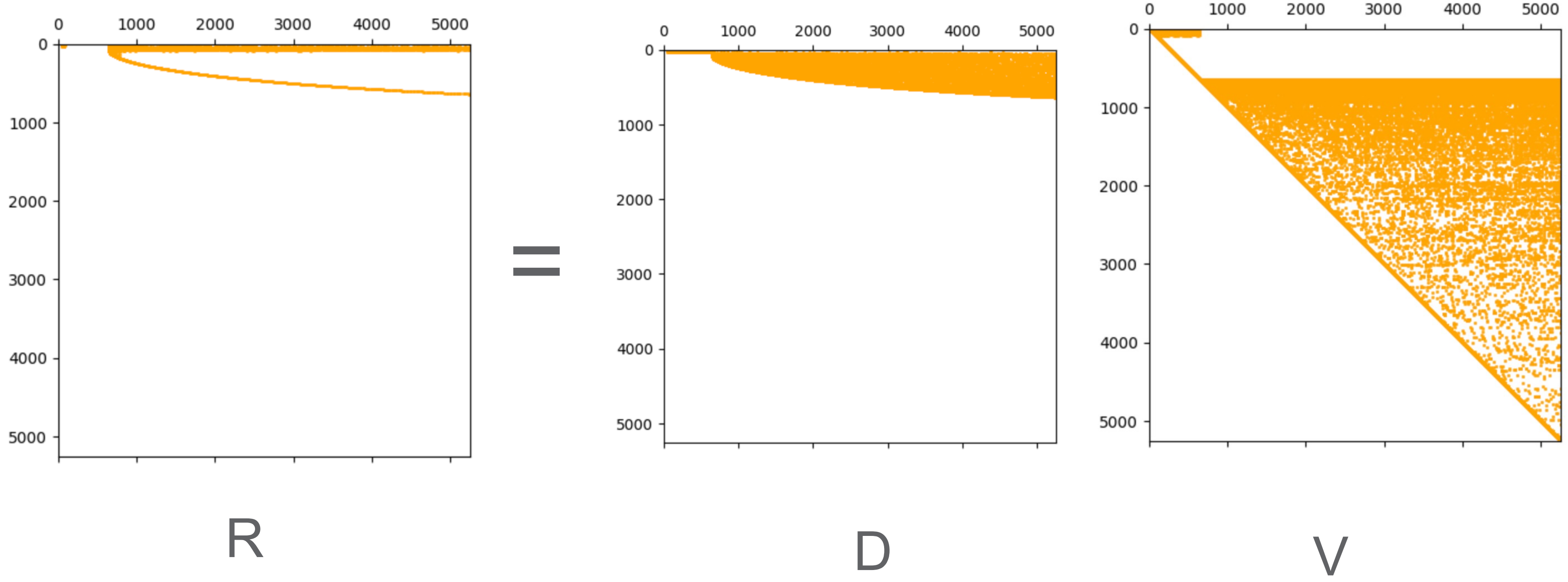


Sparse algebra



Vectors

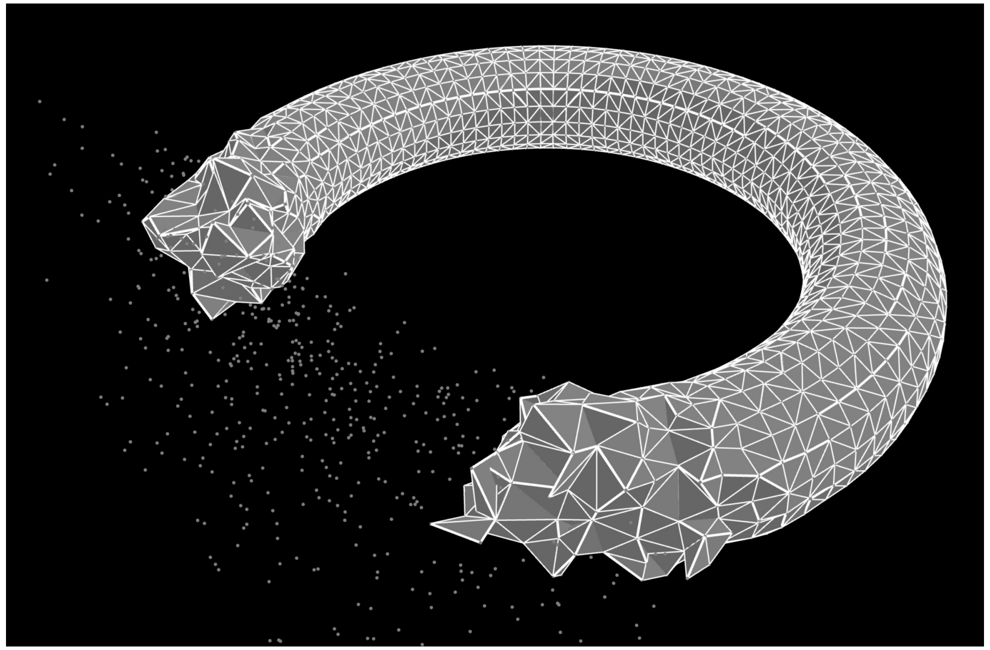
- Addition
- Linear combination
- Matrix-multiplication
- Back-substitution (triangular solve)

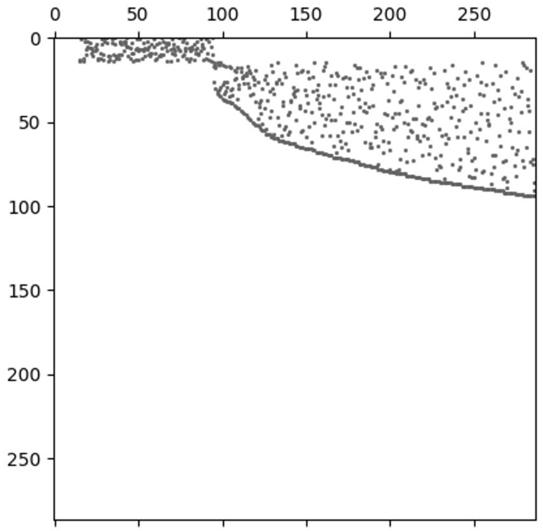


Challenges

Boundary matrices

- Billions of rows and columns
- Indexed by simplices (not integers)

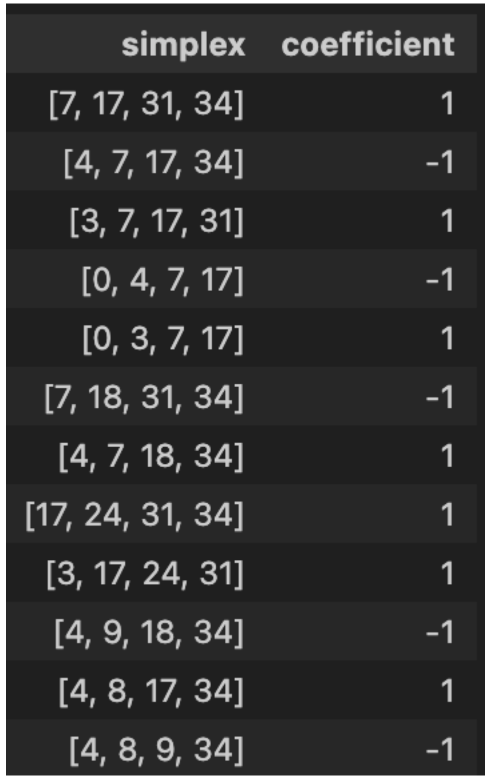




∂

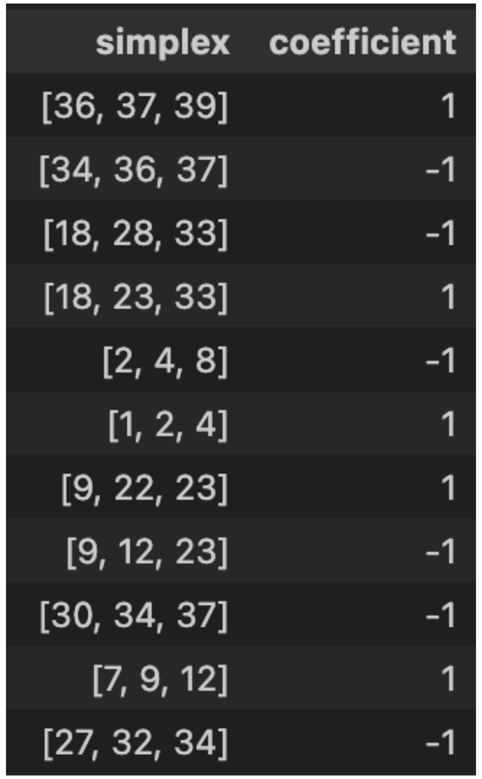
Challenges

$\partial \cdot$



simplex	coefficient
[7, 17, 31, 34]	1
[4, 7, 17, 34]	-1
[3, 7, 17, 31]	1
[0, 4, 7, 17]	-1
[0, 3, 7, 17]	1
[7, 18, 31, 34]	-1
[4, 7, 18, 34]	1
[17, 24, 31, 34]	1
[3, 17, 24, 31]	1
[4, 9, 18, 34]	-1
[4, 8, 17, 34]	1
[4, 8, 9, 34]	-1

$=$

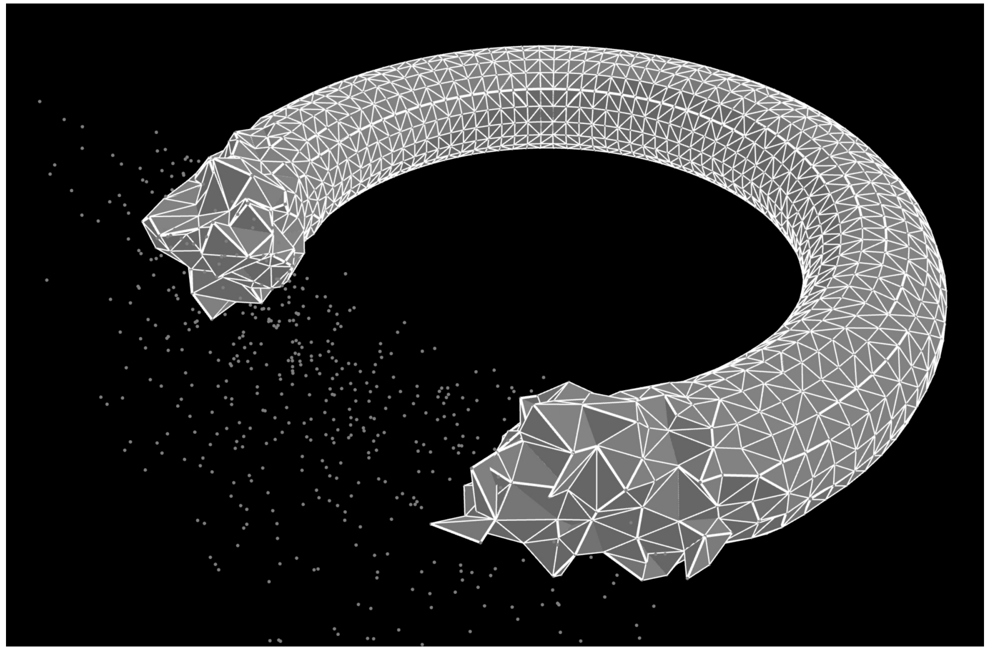


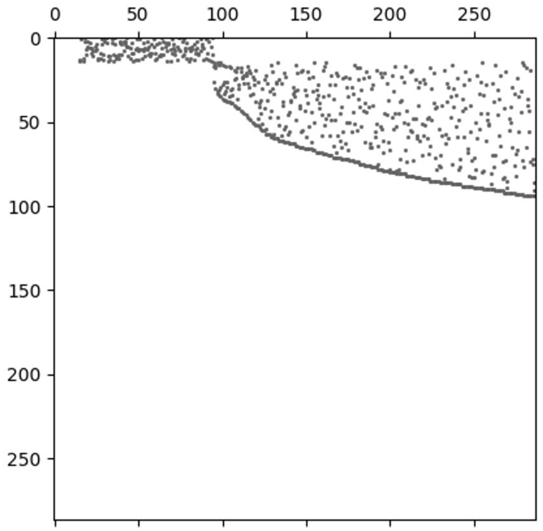
simplex	coefficient
[36, 37, 39]	1
[34, 36, 37]	-1
[18, 28, 33]	-1
[18, 23, 33]	1
[2, 4, 8]	-1
[1, 2, 4]	1
[9, 22, 23]	1
[9, 12, 23]	-1
[30, 34, 37]	-1
[7, 9, 12]	1
[27, 32, 34]	-1

Human-readable vector-matrix multiplication

Open Applied Topology

Fast, user friendly software for large boundary matrices

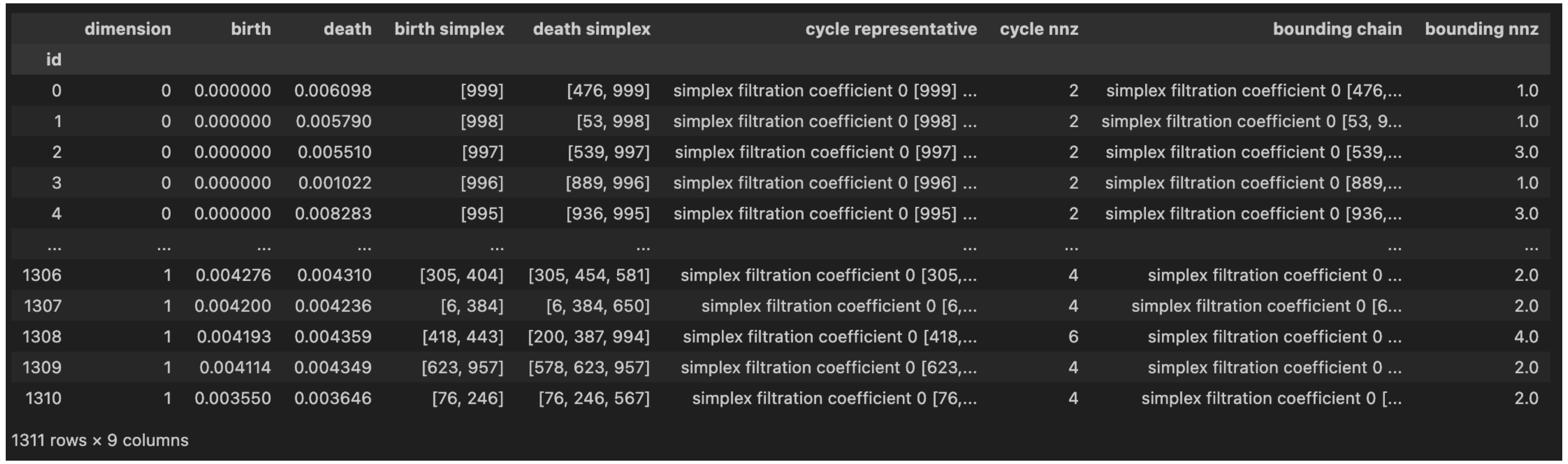




∂

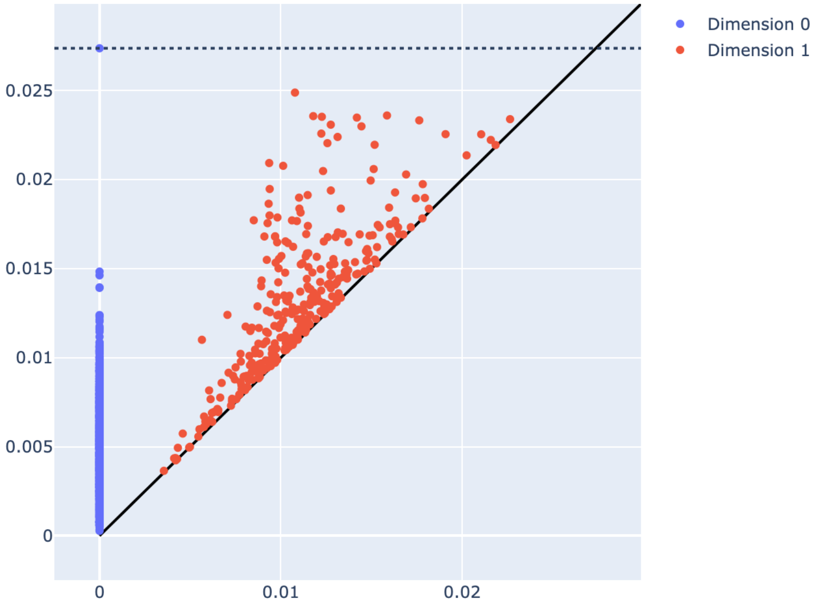
Homology

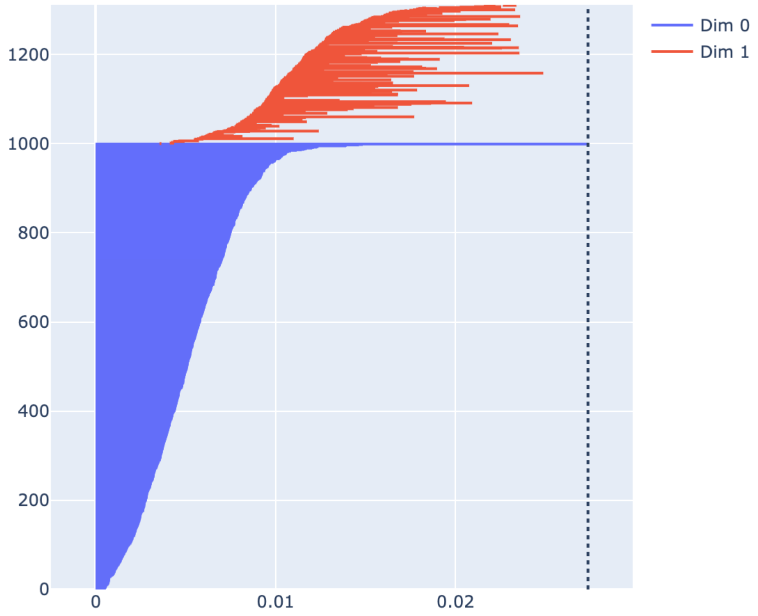
- Persistence diagrams
- Cycle representatives
- Duality
- Optimization



	dimension	birth	death	birth simplex	death simplex	cycle representative	cycle nnz	bounding chain	bounding nnz
id									
0	0	0.000000	0.006098	[999]	[476, 999]	simplex filtration coefficient 0 [999] ...	2	simplex filtration coefficient 0 [476,...	1.0
1	0	0.000000	0.005790	[998]	[53, 998]	simplex filtration coefficient 0 [998] ...	2	simplex filtration coefficient 0 [53, 9...	1.0
2	0	0.000000	0.005510	[997]	[539, 997]	simplex filtration coefficient 0 [997] ...	2	simplex filtration coefficient 0 [539,...	3.0
3	0	0.000000	0.001022	[996]	[889, 996]	simplex filtration coefficient 0 [996] ...	2	simplex filtration coefficient 0 [889,...	1.0
4	0	0.000000	0.008283	[995]	[936, 995]	simplex filtration coefficient 0 [995] ...	2	simplex filtration coefficient 0 [936,...	3.0
...	...	...	...	...	...	...	...	...	...
1306	1	0.004276	0.004310	[305, 404]	[305, 454, 581]	simplex filtration coefficient 0 [305,...	4	simplex filtration coefficient 0 ...	2.0
1307	1	0.004200	0.004236	[6, 384]	[6, 384, 650]	simplex filtration coefficient 0 [6,...	4	simplex filtration coefficient 0 [6...	2.0
1308	1	0.004193	0.004359	[418, 443]	[200, 387, 994]	simplex filtration coefficient 0 [418,...	6	simplex filtration coefficient 0 ...	4.0
1309	1	0.004114	0.004349	[623, 957]	[578, 623, 957]	simplex filtration coefficient 0 [623,...	4	simplex filtration coefficient 0 ...	2.0
1310	1	0.003550	0.003646	[76, 246]	[76, 246, 567]	simplex filtration coefficient 0 [76,...	4	simplex filtration coefficient 0 [...	2.0

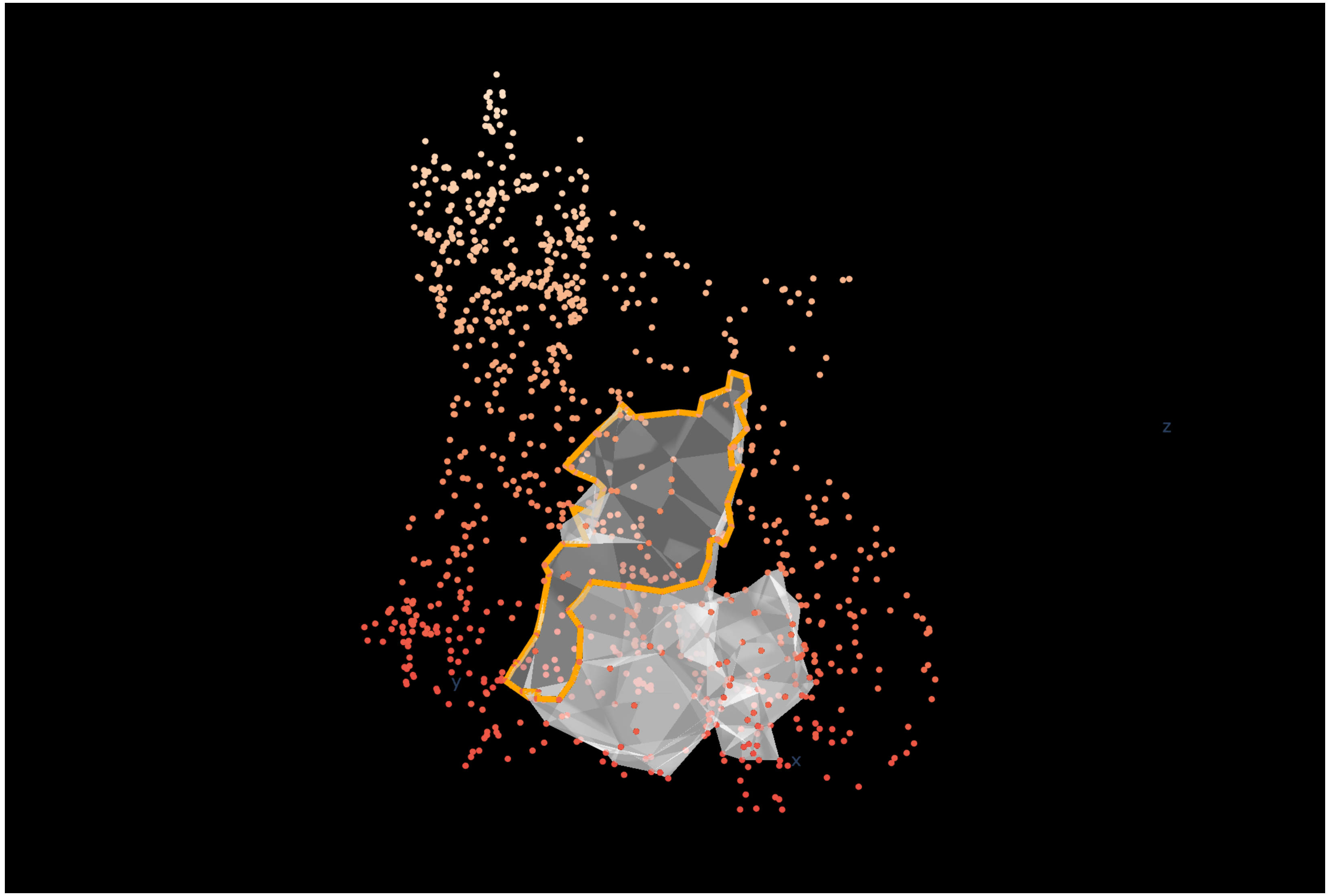
1311 rows × 9 columns





Homology

- Persistence diagrams
- Cycle representatives
- Duality
- Optimization

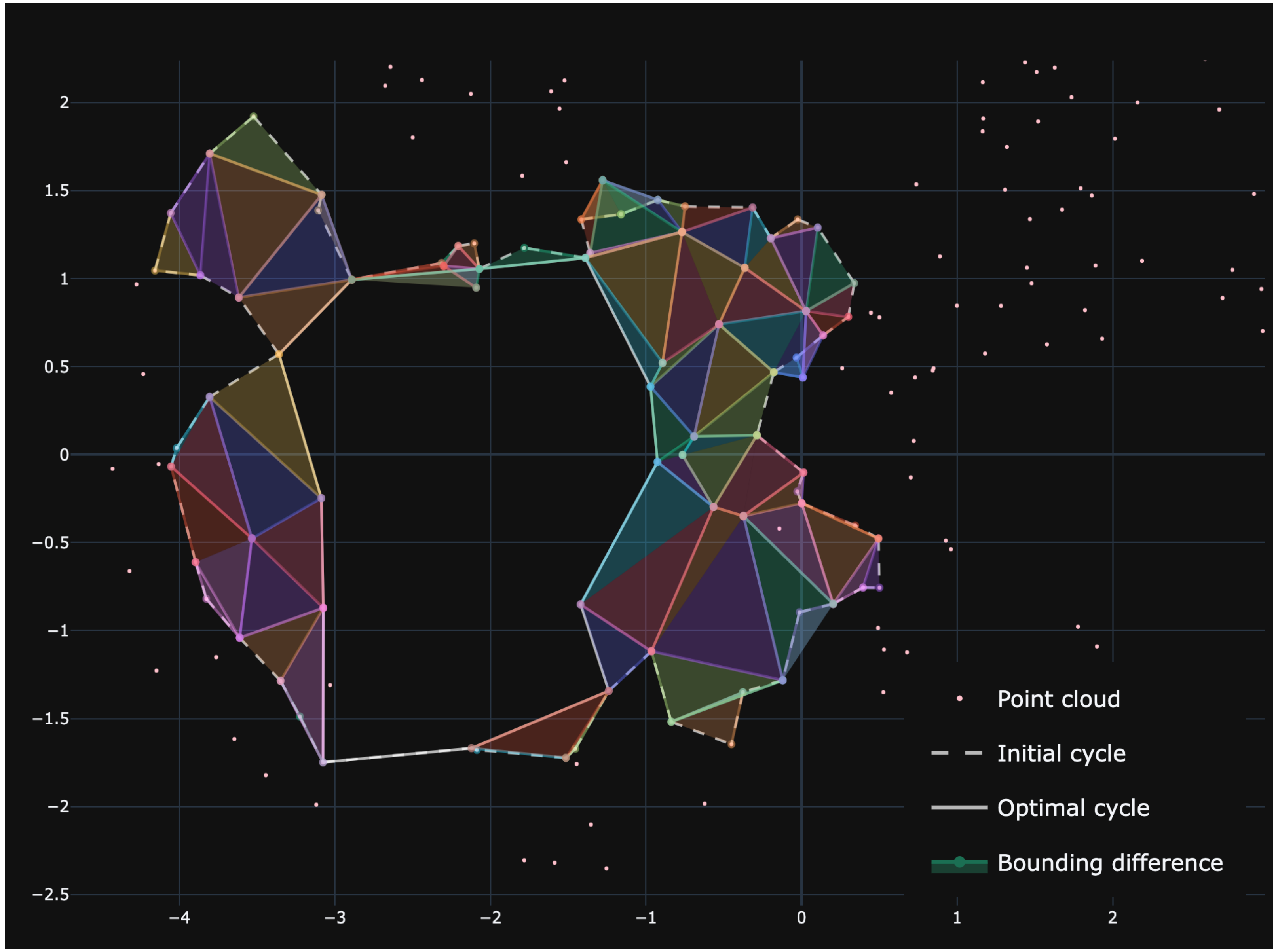


Stanford Dragon
Persistent cycle representative (orange)
Bounding chain (white)

Pacific Northwest
NATIONAL LABORATORY

Homology

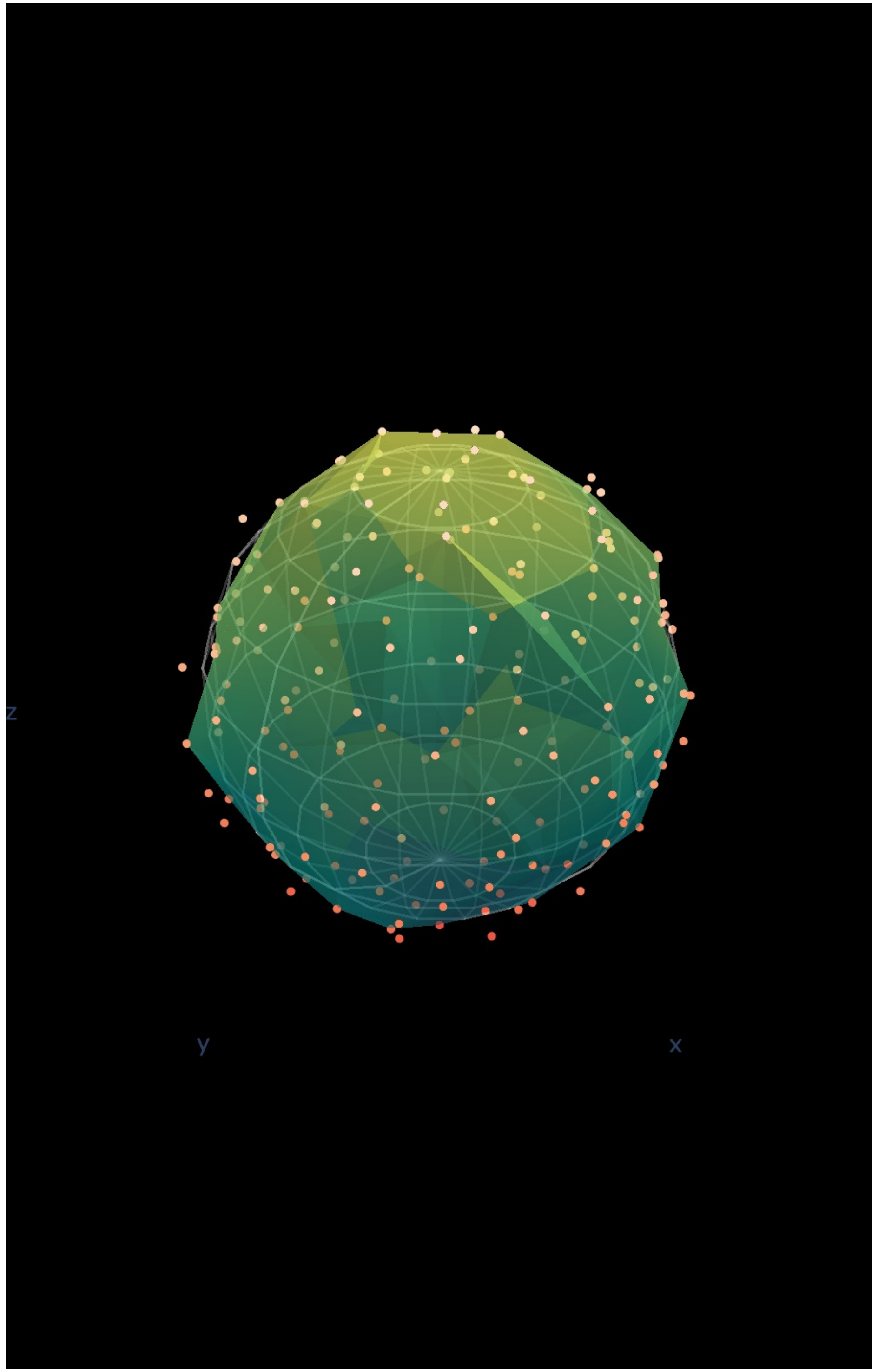
- Persistence diagrams
- Cycle representatives
- Duality
- Optimization



Localized cycle

Packages

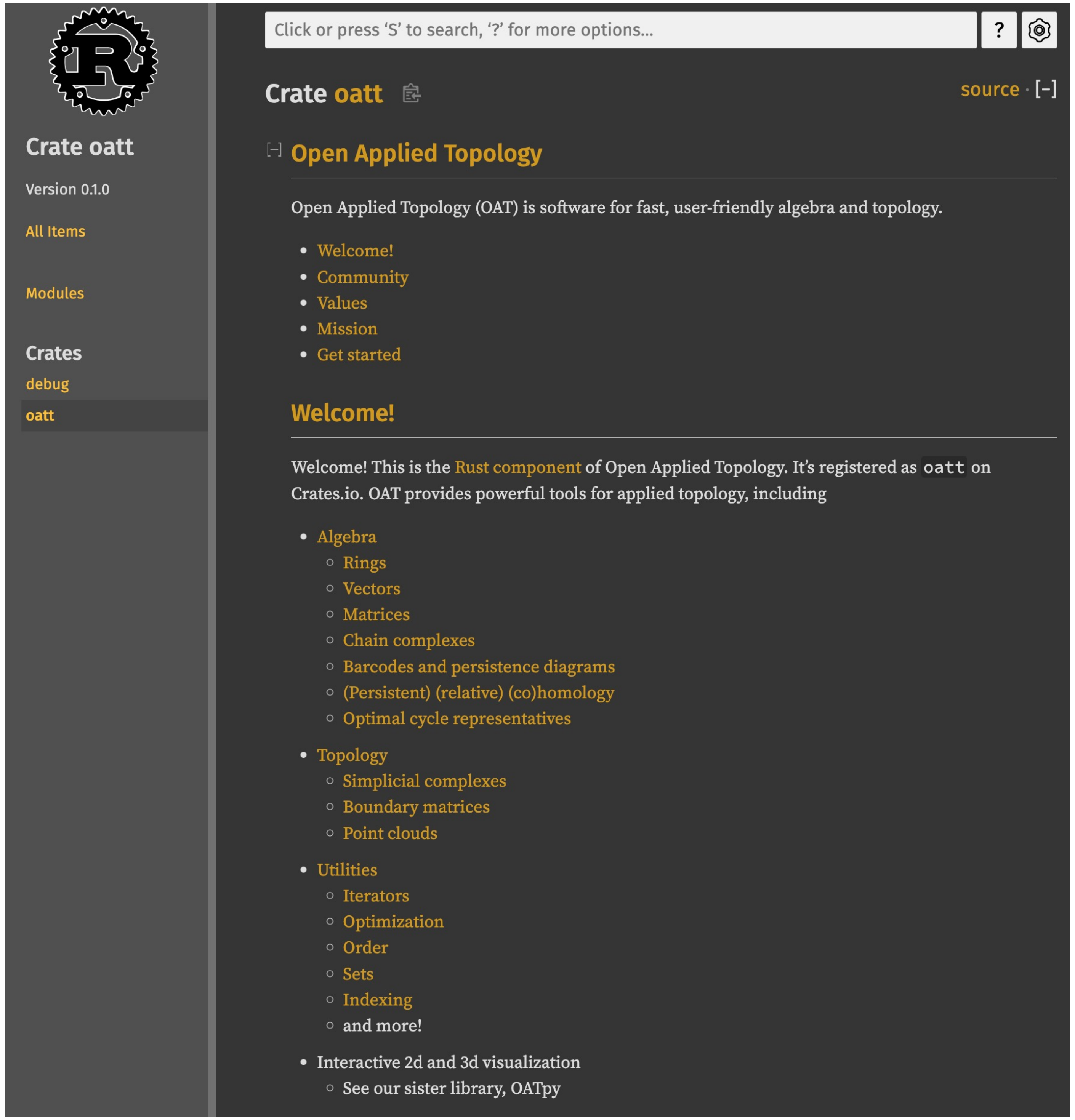
- Rust
  - Low-level implementation
- Python
  - User-friendly interface
- Notebooks
  - Python + Rust tutorials
- Executable
  - Available for all Rust programs



Pacific Northwest
NATIONAL LABORATORY

Packages

- Rust
  - Low-level implementation
- Python
  - User-friendly interface
- Notebooks
  - Python + Rust tutorials
- Executable
  - Available for all Rust programs



Crate oatt

Open Applied Topology

Open Applied Topology (OAT) is software for fast, user-friendly algebra and topology.

- Welcome!
- Community
- Values
- Mission
- Get started

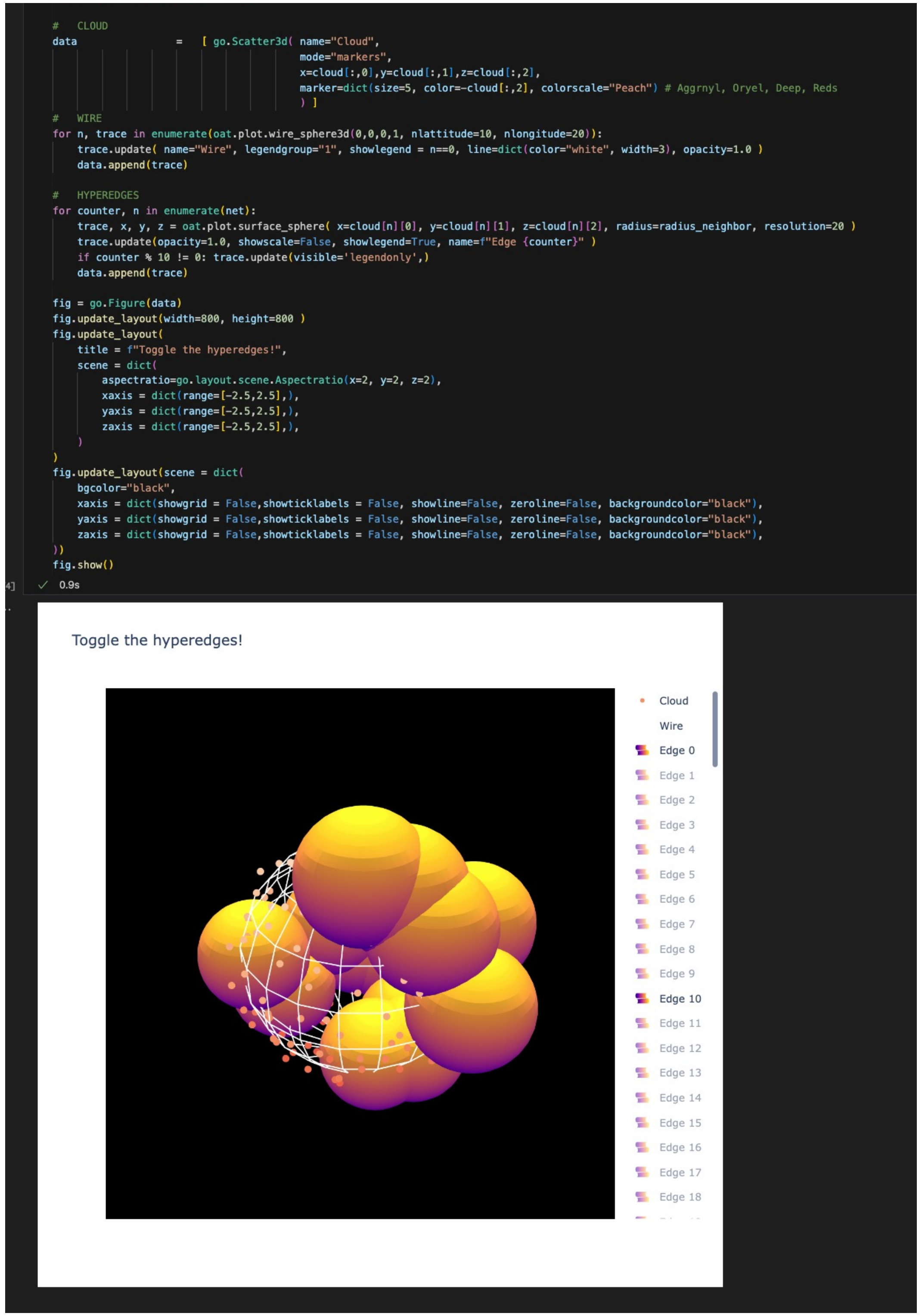
Welcome!

Welcome! This is the Rust component of Open Applied Topology. It's registered as `oatt` on Crates.io. OAT provides powerful tools for applied topology, including

- Algebra
  - Rings
  - Vectors
  - Matrices
  - Chain complexes
  - Barcodes and persistence diagrams
  - (Persistent) (relative) (co)homology
  - Optimal cycle representatives
- Topology
  - Simplicial complexes
  - Boundary matrices
  - Point clouds
- Utilities
  - Iterators
  - Optimization
  - Order
  - Sets
  - Indexing
  - and more!
- Interactive 2d and 3d visualization
  - See our sister library, OATpy

Packages

- Rust
  - Low-level implementation
- **Python**
  - **User-friendly interface**
- **Notebooks**
  - **Python + Rust tutorials**
- Executable
  - Available for all Rust programs



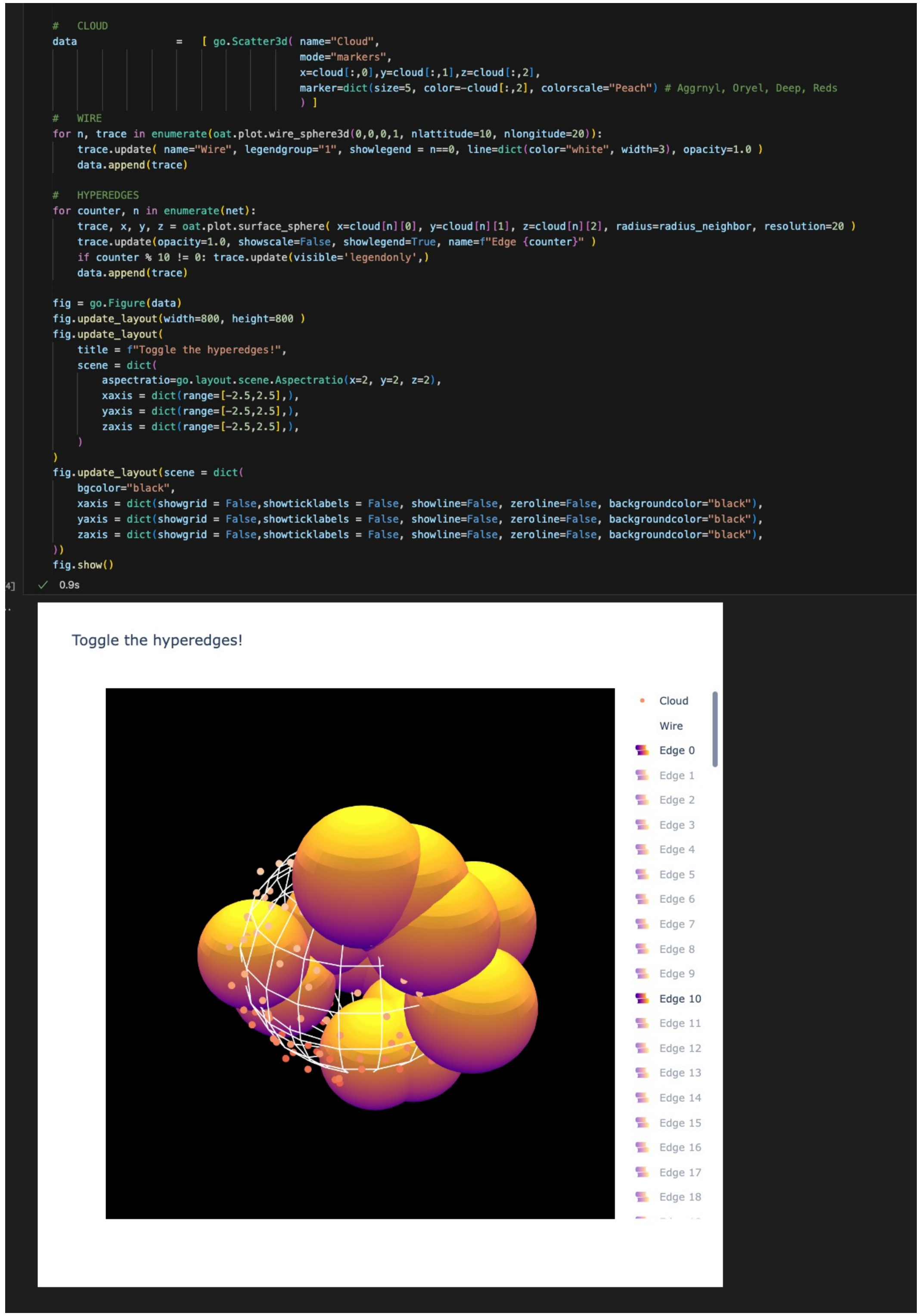
```python
#   CLOUD
data                =   [ go.Scatter3d( name="Cloud",
                                        mode="markers",
                                        x=cloud[:,0],y=cloud[:,1],z=cloud[:,2],
                                        marker=dict(size=5, color=-cloud[:,2], colorscale="Peach") # Aggrnyl, Oryel, Deep, Reds
                                        ) ]
#   WIRE
for n, trace in enumerate(oat.plot.wire_sphere3d(0,0,0,1, nlattitude=10, nlongitude=20)):
    trace.update( name="Wire", legendgroup="1", showlegend = n==0, line=dict(color="white", width=3), opacity=1.0 )
    data.append(trace)

#   HYPEREDGES
for counter, n in enumerate(net):
    trace, x, y, z = oat.plot.surface_sphere( x=cloud[n][0], y=cloud[n][1], z=cloud[n][2], radius=radius_neighbor, resolution=20 )
    trace.update(opacity=1.0, showscale=False, showlegend=True, name=f"Edge {counter}" )
    if counter % 10 != 0: trace.update(visible='legendonly',)
    data.append(trace)

fig = go.Figure(data)
fig.update_layout(width=800, height=800 )
fig.update_layout(
    title = f"Toggle the hyperedges!",
    scene = dict(
        aspectratio=go.layout.scene.Aspectratio(x=2, y=2, z=2),
        xaxis = dict(range=[-2.5,2.5],),
        yaxis = dict(range=[-2.5,2.5],),
        zaxis = dict(range=[-2.5,2.5],),
    )
)
fig.update_layout(scene = dict(
    bgcolor="black",
    xaxis = dict(showgrid = False,showticklabels = False, showline=False, zeroline=False, backgroundcolor="black"),
    yaxis = dict(showgrid = False,showticklabels = False, showline=False, zeroline=False, backgroundcolor="black"),
    zaxis = dict(showgrid = False,showticklabels = False, showline=False, zeroline=False, backgroundcolor="black"),
))
fig.show()
```

✓ 0.9s

Toggle the hyperedges!

Packages

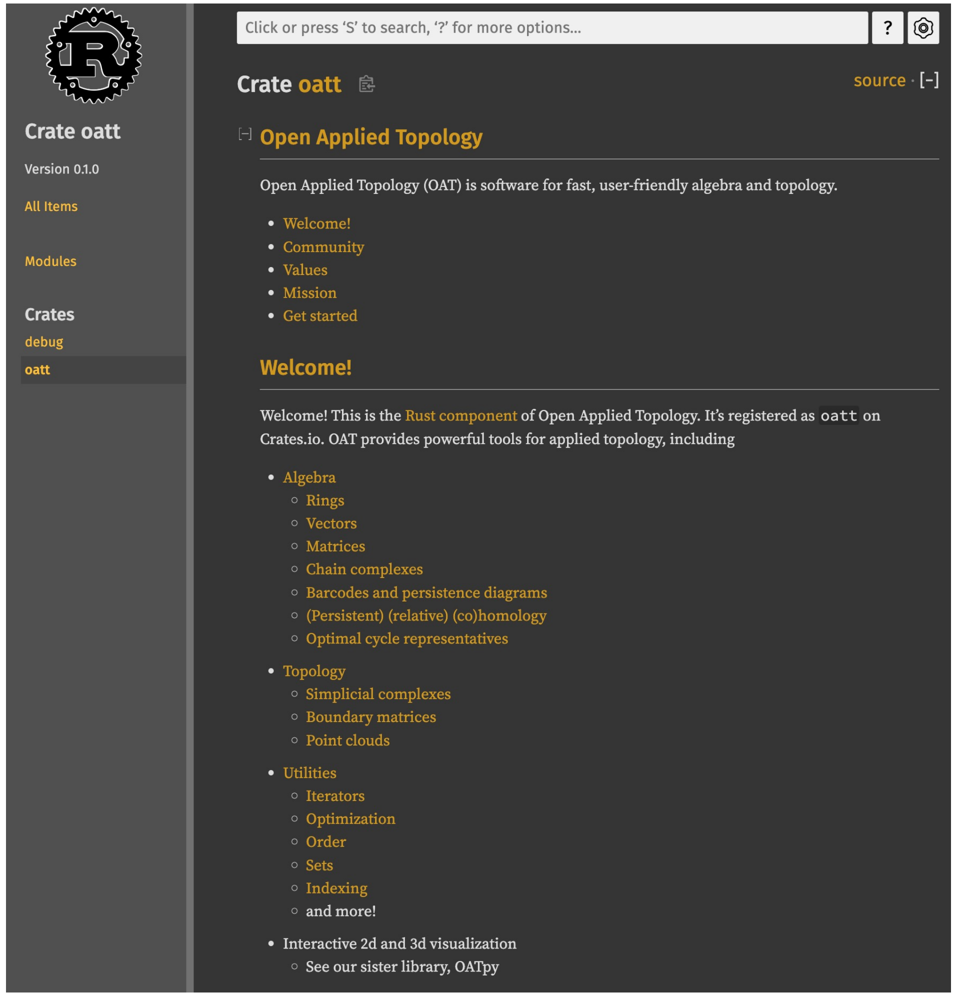
- Rust
  - Low-level implementation
- Python
  - User-friendly interface
- Notebooks
  - Python + Rust tutorials
- **Executable**
  - **Available for all Rust programs**



```python
#   CLOUD
data                =   [ go.Scatter3d( name="Cloud",
                                        mode="markers",
                                        x=cloud[:,0],y=cloud[:,1],z=cloud[:,2],
                                        marker=dict(size=5, color=-cloud[:,2], colorscale="Peach") # Aggrnyl, Oryel, Deep, Reds
                                        ) ]
#   WIRE
for n, trace in enumerate(oat.plot.wire_sphere3d(0,0,0,1, nlattitude=10, nlongitude=20)):
    trace.update( name="Wire", legendgroup="1", showlegend = n==0, line=dict(color="white", width=3), opacity=1.0 )
    data.append(trace)

#   HYPEREDGES
for counter, n in enumerate(net):
    trace, x, y, z = oat.plot.surface_sphere( x=cloud[n][0], y=cloud[n][1], z=cloud[n][2], radius=radius_neighbor, resolution=20 )
    trace.update(opacity=1.0, showscale=False, showlegend=True, name=f"Edge {counter}" )
    if counter % 10 != 0: trace.update(visible='legendonly',)
    data.append(trace)

fig = go.Figure(data)
fig.update_layout(width=800, height=800 )
fig.update_layout(
    title = f"Toggle the hyperedges!",
    scene = dict(
        aspectratio=go.layout.scene.Aspectratio(x=2, y=2, z=2),
        xaxis = dict(range=[-2.5,2.5],),
        yaxis = dict(range=[-2.5,2.5],),
        zaxis = dict(range=[-2.5,2.5],),
    )
)
fig.update_layout(scene = dict(
    bgcolor="black",
    xaxis = dict(showgrid = False,showticklabels = False, showline=False, zeroline=False, backgroundcolor="black"),
    yaxis = dict(showgrid = False,showticklabels = False, showline=False, zeroline=False, backgroundcolor="black"),
    zaxis = dict(showgrid = False,showticklabels = False, showline=False, zeroline=False, backgroundcolor="black"),
))
fig.show()
```

```
✓ 0.9s
```

Toggle the hyperedges!



Crate oatt

Open Applied Topology

Open Applied Topology (OAT) is software for fast, user-friendly algebra and topology.

- Welcome!
- Community
- Values
- Mission
- Get started

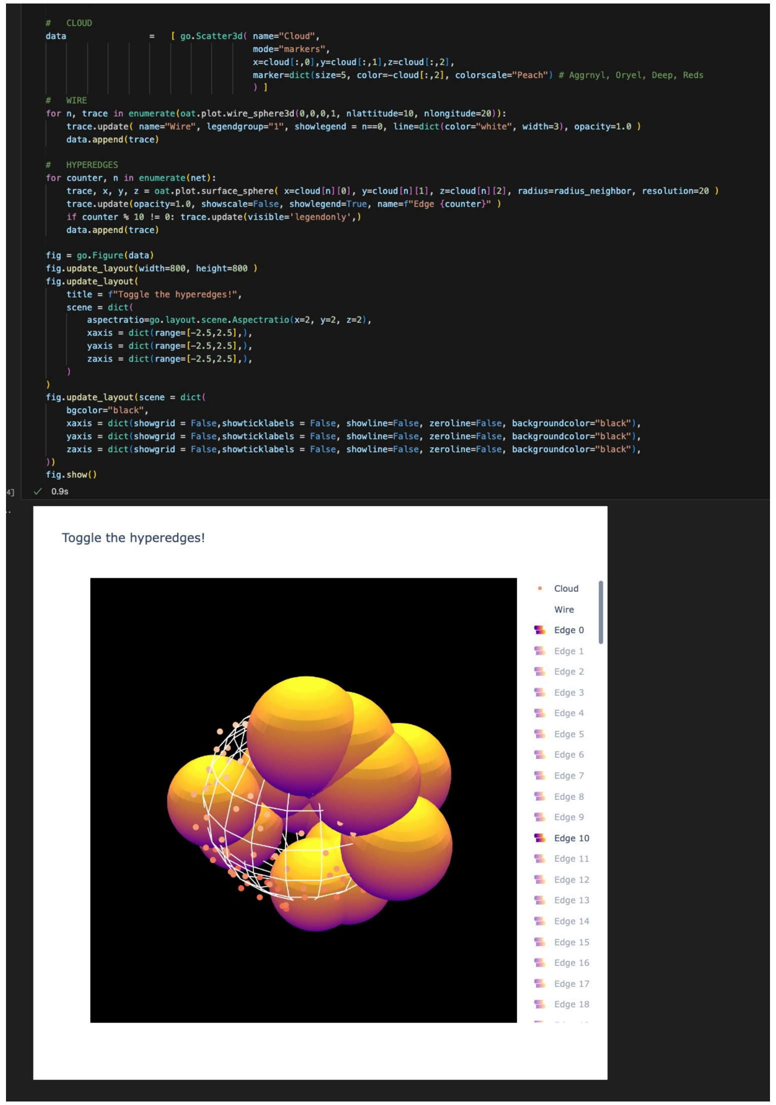
Welcome!

Welcome! This is the Rust component of Open Applied Topology. It's registered as `oatt` on Crates.io. OAT provides powerful tools for applied topology, including

- Algebra
  - Rings
  - Vectors
  - Matrices
  - Chain complexes
  - Barcodes and persistence diagrams
  - (Persistent) (relative) (co)homology
  - Optimal cycle representatives
- Topology
  - Simplicial complexes
  - Boundary matrices
  - Point clouds
- Utilities
  - Iterators
  - Optimization
  - Order
  - Sets
  - Indexing
  - and more!
- Interactive 2d and 3d visualization
  - See our sister library, OATpy

Open Applied Topology

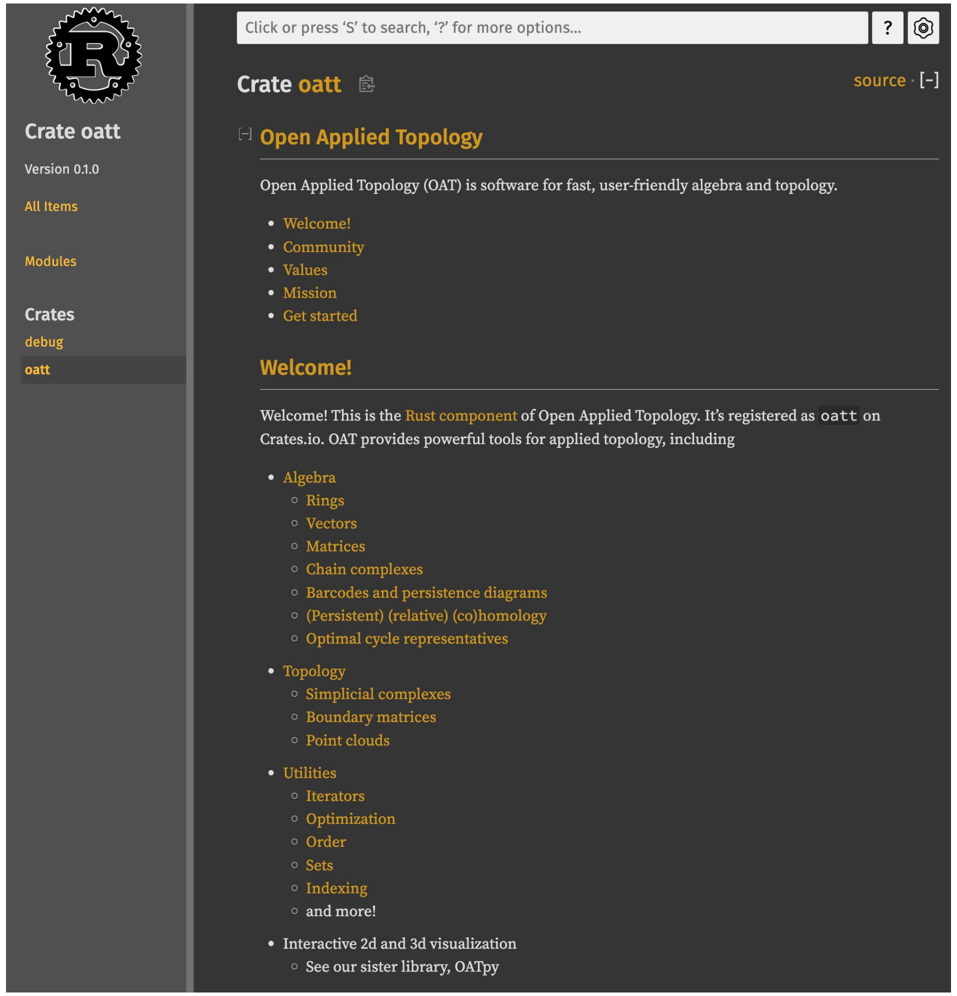
- Performance
- Accessibility
- Modularity



```
#   CLOUD
data                =   [ go.Scatter3d( name="Cloud",
                                        mode="markers",
                                        x=cloud[:,0],y=cloud[:,1],z=cloud[:,2],
                                        marker=dict(size=5, color=-cloud[:,2], colorscale="Peach") # Aggrnyl, Oryel, Deep, Reds
                                        ) ]
#   WIRE
for n, trace in enumerate(oat.plot.wire_sphere3d(0,0,0,1, nlattitude=10, nlongitude=20)):
    trace.update( name="Wire", legendgroup="1", showlegend = n==0, line=dict(color="white", width=3), opacity=1.0 )
    data.append(trace)

#   HYPEREDGES
for counter, n in enumerate(net):
    trace, x, y, z = oat.plot.surface_sphere( x=cloud[n][0], y=cloud[n][1], z=cloud[n][2], radius=radius_neighbor, resolution=20 )
    trace.update(opacity=1.0, showscale=False, showlegend=True, name=f"Edge {counter}" )
    if counter % 10 != 0: trace.update(visible='legendonly',)
    data.append(trace)

fig = go.Figure(data)
fig.update_layout(width=800, height=800 )
fig.update_layout(
    title = f"Toggle the hyperedges!",
    scene = dict(
        aspectratio=go.layout.scene.Aspectratio(x=2, y=2, z=2),
        xaxis = dict(range=[-2.5,2.5],),
        yaxis = dict(range=[-2.5,2.5],),
        zaxis = dict(range=[-2.5,2.5],),
    )
)
fig.update_layout(scene = dict(
    bgcolor="black",
    xaxis = dict(showgrid = False,showticklabels = False, showline=False, zeroline=False, backgroundcolor="black"),
    yaxis = dict(showgrid = False,showticklabels = False, showline=False, zeroline=False, backgroundcolor="black"),
    zaxis = dict(showgrid = False,showticklabels = False, showline=False, zeroline=False, backgroundcolor="black"),
))
fig.show()
```

Open Applied Topology

- Performance
- Accessibility
- Modularity



Crate oatt

Open Applied Topology

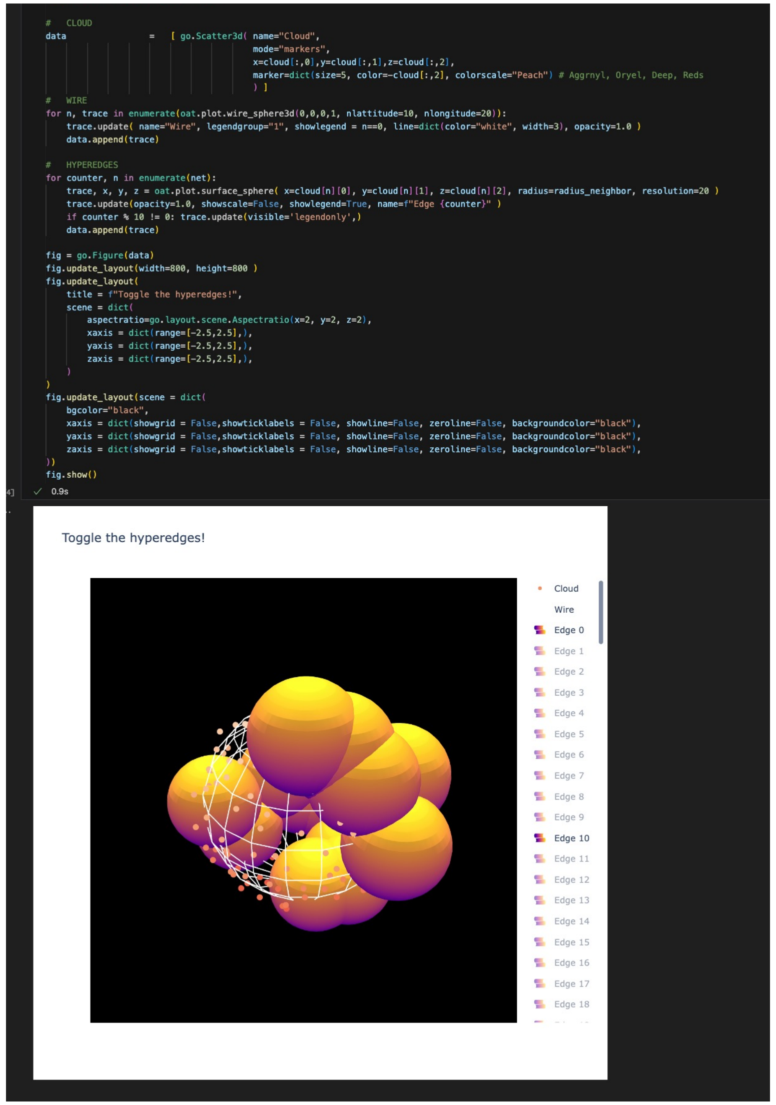
Open Applied Topology (OAT) is software for fast, user-friendly algebra and topology.

- Welcome!
- Community
- Values
- Mission
- Get started

Welcome!

Welcome! This is the Rust component of Open Applied Topology. It's registered as `oatt` on Crates.io. OAT provides powerful tools for applied topology, including

- Algebra
  - Rings
  - Vectors
  - Matrices
  - Chain complexes
  - Barcodes and persistence diagrams
  - (Persistent) (relative) (co)homology
  - Optimal cycle representatives
- Topology
  - Simplicial complexes
  - Boundary matrices
  - Point clouds
- Utilities
  - Iterators
  - Optimization
  - Order
  - Sets
  - Indexing
  - and more!
- Interactive 2d and 3d visualization
  - See our sister library, OATpy



```
#   CLOUD
data                =   [ go.Scatter3d( name="Cloud",
                                        mode="markers",
                                        x=cloud[:,0],y=cloud[:,1],z=cloud[:,2],
                                        marker=dict(size=5, color=-cloud[:,2], colorscale="Peach") # Aggrnyl, Oryel, Deep, Reds
                                        ) ]
#   WIRE
for n, trace in enumerate(oat.plot.wire_sphere3d(0,0,0,1, nlattitude=10, nlongitude=20)):
    trace.update( name="Wire", legendgroup="1", showlegend = n==0, line=dict(color="white", width=3), opacity=1.0 )
    data.append(trace)

#   HYPEREDGES
for counter, n in enumerate(net):
    trace, x, y, z = oat.plot.surface_sphere( x=cloud[n][0], y=cloud[n][1], z=cloud[n][2], radius=radius_neighbor, resolution=20 )
    trace.update(opacity=1.0, showscale=False, showlegend=True, name=f"Edge {counter}" )
    if counter % 10 != 0: trace.update(visible='legendonly',)
    data.append(trace)

fig = go.Figure(data)
fig.update_layout(width=800, height=800 )
fig.update_layout(
    title = f"Toggle the hyperedges!",
    scene = dict(
        aspectratio=go.layout.scene.Aspectratio(x=2, y=2, z=2),
        xaxis = dict(range=[-2.5,2.5],),
        yaxis = dict(range=[-2.5,2.5],),
        zaxis = dict(range=[-2.5,2.5],),
    )
)
fig.update_layout(scene = dict(
    bgcolor="black",
    xaxis = dict(showgrid = False,showticklabels = False, showline=False, zeroline=False, backgroundcolor="black"),
    yaxis = dict(showgrid = False,showticklabels = False, showline=False, zeroline=False, backgroundcolor="black"),
    zaxis = dict(showgrid = False,showticklabels = False, showline=False, zeroline=False, backgroundcolor="black"),
))
fig.show()
```

Support

- ONR N00014-16-1-2010
- NSF-1934960
- NSF DMS-1854748
- EP/R018472/1
- Swartz Center for Theoretical Neuroscience, Princeton University

Pacific Northwest
NATIONAL LABORATORY

BROWN

intel

MACALESTER

MICHIGAN STATE
UNIVERSITY

AFRL
THE AIR FORCE RESEARCH LABORATORY
LEAD | DISCOVER | DEVELOP | DELIVER

Penn
UNIVERSITY of PENNSYLVANIA

PRINCETON
UNIVERSITY

ST. OLAF
COLLEGE

UALBANY

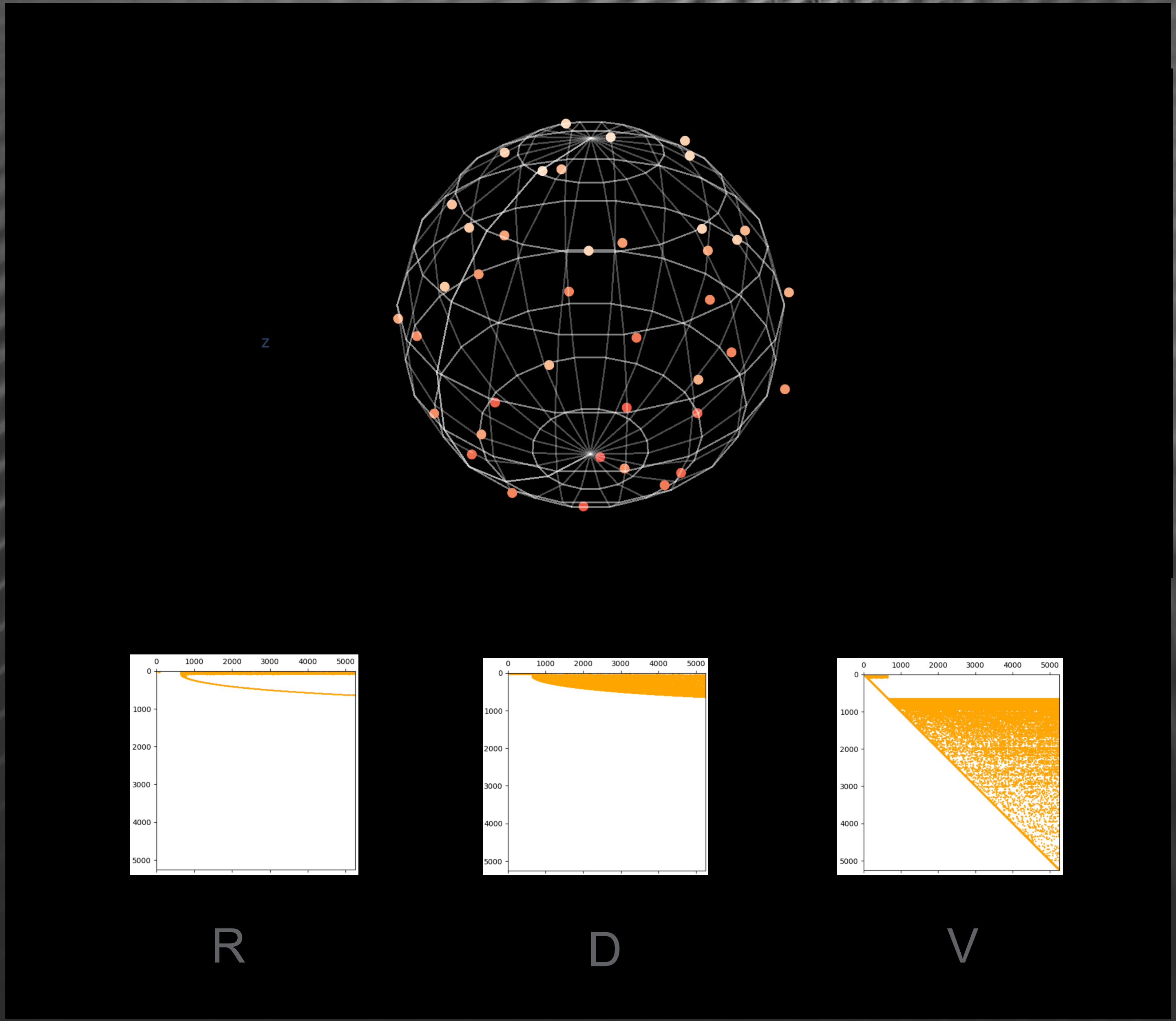
UNIVERSITY OF
DELAWARE

UNIVERSITY OF
OXFORD

Pacific Northwest
NATIONAL LABORATORY

Open Applied Topology

Available at
https://openappliedtopology.github.io



U.S. DEPARTMENT OF ENERGY
BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

PNNL is operated by Battelle for the U.S. Department of Energy