3D Volume Plots in Python

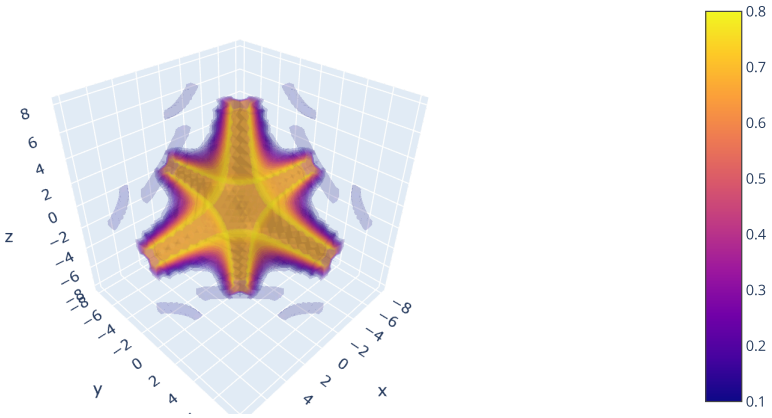
How to make 3D Volume Plots in Python with Plotly.

A volume plot with `go.Volume` shows several partially transparent isosurfaces for volume rendering. The API of `go.Volume` is close to the one of `go.Isosurface`. However, whereas isosurface plots (/python/3d-isosurface-plots/) show all surfaces with the same opacity, tweaking the `opacityscale` parameter of `go.Volume` results in a depth effect and better volume rendering.

Simple volume plot with go.Volume

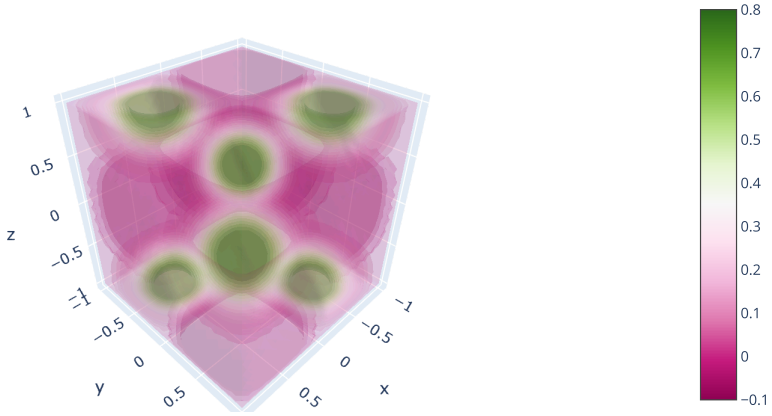
In the three examples below, note that the default colormap is different whether isomin and isomax have the same sign or not.

```
import plotly.graph_objects as go
import numpy as np
X, Y, Z = np.mgrid[-8:8:40j, -8:8:40j, -8:8:40j]
values = np.sin(X*Y*Z) / (X*Y*Z)

fig = go.Figure(data=go.Volume(
    x=X.flatten(),
    y=Y.flatten(),
    z=Z.flatten(),
    value=values.flatten(),
    isomin=0.1,
    isomax=0.8,
    opacity=0.1, # needs to be small to see through all surfaces
    surface_count=17, # needs to be a large number for good volume rendering
    ))
fig.show()
```



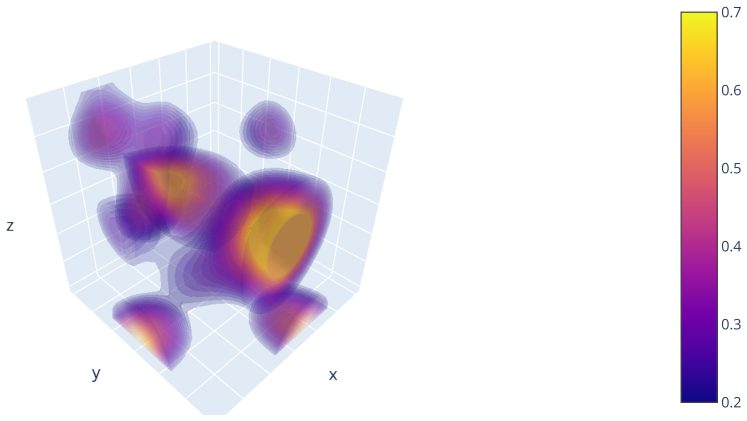
e plots

```
import plotly.graph_objects as go
import numpy as np
X, Y, Z = np.mgrid[-1:1:30j, -1:1:30j, -1:1:30j]
values =    np.sin(np.pi*X) * np.cos(np.pi*Z) * np.sin(np.pi*Y)

fig = go.Figure(data=go.Volume(
    x=X.flatten(),
    y=Y.flatten(),
    z=Z.flatten(),
    value=values.flatten(),
    isomin=-0.1,
    isomax=0.8,
    opacity=0.1, # needs to be small to see through all surfaces
    surface_count=21, # needs to be a large number for good volume rendering
    ))
fig.show()
```



e plots

```
import numpy as np
import plotly.graph_objects as go

# Generate nicely looking random 3D-field
np.random.seed(0)
l = 30
X, Y, Z = np.mgrid[:l, :l, :l]
vol = np.zeros((l, l, l))
pts = (l * np.random.rand(3, 15)).astype(int)
vol[tuple(indices for indices in pts)] = 1
from scipy import ndimage
vol = ndimage.gaussian_filter(vol, 4)
vol /= vol.max()

fig = go.Figure(data=go.Volume(
    x=X.flatten(), y=Y.flatten(), z=Z.flatten(),
    value=vol.flatten(),
    isomin=0.2,
    isomax=0.7,
    opacity=0.1,
    surface_count=25,
    ))
fig.update_layout(scene_xaxis_showticklabels=False,
                  scene_yaxis_showticklabels=False,
                  scene_zaxis_showticklabels=False)
fig.show()
```

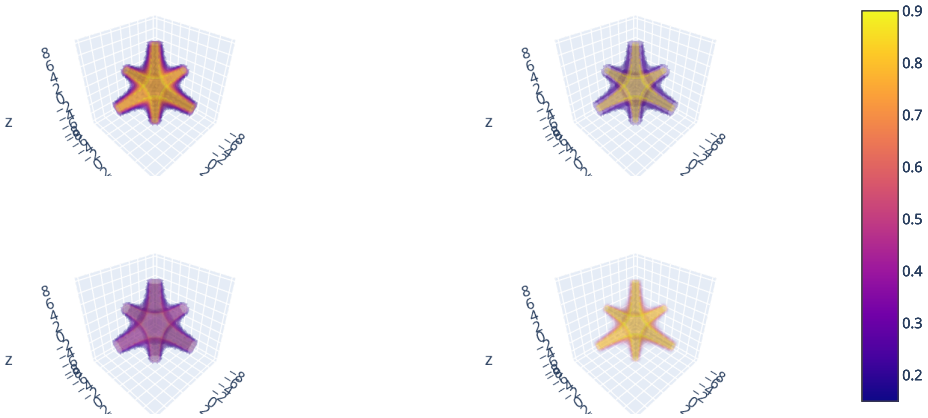


Defining the opacity scale of volume plots

In order to see through the volume, the different isosurfaces need to be partially transparent. This transparency is controlled by a global parameter, opacity, as well as an opacity scale mapping scalar values to opacity levels. The figure below shows that changing the opacity scale changes a lot the visualization, so that opacityscale should be chosen carefully (uniform corresponds to a uniform opacity, min/max maps the minimum/maximum value to a maximal opacity, and extremes maps both the minimum and maximum values to maximal opacity, with a dip in between).

e plots

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots
fig = make_subplots(
    rows=2, cols=2,
    specs=[[{'type': 'volume'}, {'type': 'volume'}],
           [{'type': 'volume'}, {'type': 'volume'}]])

import numpy as np

X, Y, Z = np.mgrid[-8:8:30j, -8:8:30j, -8:8:30j]
values =    np.sin(X*Y*Z) / (X*Y*Z)


fig.add_trace(go.Volume(
    opacityscale="uniform",
    ), row=1, col=1)
fig.add_trace(go.Volume(
    opacityscale="extremes",
    ), row=1, col=2)
fig.add_trace(go.Volume(
    opacityscale="min",
    ), row=2, col=1)
fig.add_trace(go.Volume(
    opacityscale="max",
    ), row=2, col=2)
fig.update_traces(x=X.flatten(), y=Y.flatten(), z=Z.flatten(), value=values.flatten(),
    isomin=0.15, isomax=0.9, opacity=0.1, surface_count=15)
fig.show()
```

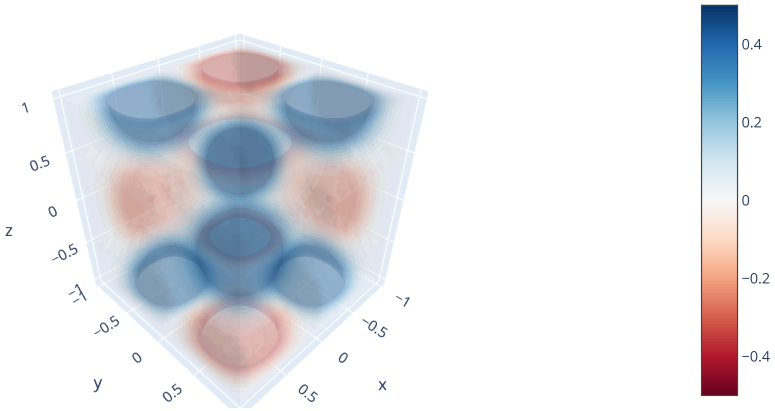


Defining a custom opacity scale

It is also possible to define a custom opacity scale, mapping scalar values to relative opacity values (between 0 and 1, the maximum opacity is given by the opacity keyword). This is useful to make a range of values completely transparent, as in the example below between -0.2 and 0.2.

e plots

```
import plotly.graph_objects as go
import numpy as np
X, Y, Z = np.mgrid[-1:1:30j, -1:1:30j, -1:1:30j]
values =    np.sin(np.pi*X) * np.cos(np.pi*Z) * np.sin(np.pi*Y)

fig = go.Figure(data=go.Volume(
    x=X.flatten(),
    y=Y.flatten(),
    z=Z.flatten(),
    value=values.flatten(),
    isomin=-0.5,
    isomax=0.5,
    opacity=0.1, # max opacity
    opacityscale=[[-0.5, 1], [-0.2, 0], [0.2, 0], [0.5, 1]],
    surface_count=21,
    colorscale='RdBu'
    ))
fig.show()
```

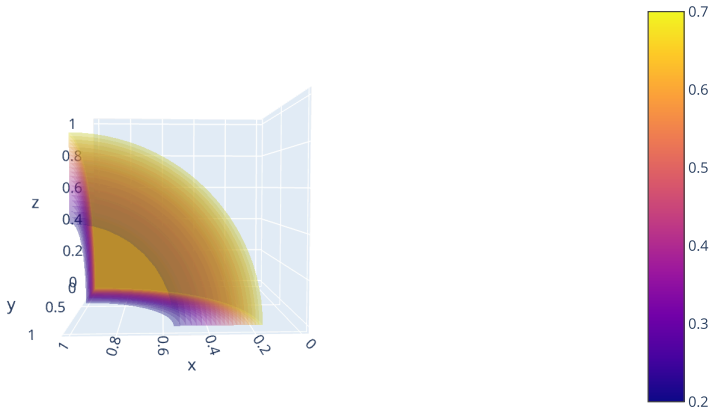


Adding caps to a volume plot

For a clearer visualization of internal surfaces, it is possible to remove the caps (color-coded surfaces on the sides of the visualization domain). Caps are visible by default. Compare below with and without caps.

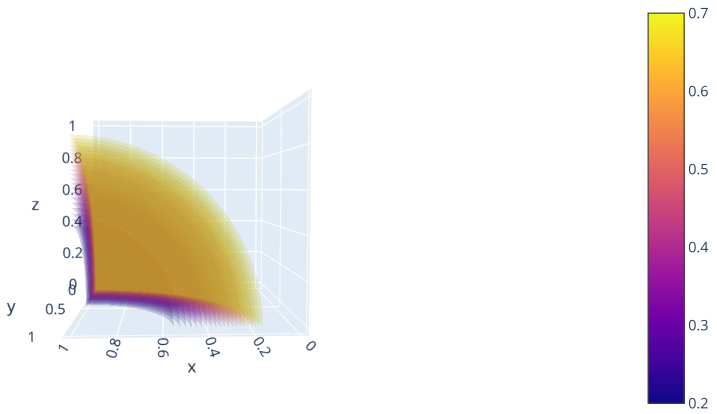
e plots

```
import numpy as np
import plotly.graph_objects as go


X, Y, Z = np.mgrid[:1:20j, :1:20j, :1:20j]
vol = (X - 1)**2 + (Y - 1)**2 + Z**2


fig = go.Figure(data=go.Volume(
    x=X.flatten(), y=Y.flatten(), z=Z.flatten(),
    value=vol.flatten(),
    isomin=0.2,
    isomax=0.7,
    opacity=0.2,
    surface_count=21,
    caps= dict(x_show=True, y_show=True, z_show=True, x_fill=1), # with caps (default mode)
    ))

# Change camera view for a better view of the sides, XZ plane
# (see https://plotly.com/python/v3/3d-camera-controls/)
fig.update_layout(scene_camera = dict(
    up=dict(x=0, y=0, z=1),
    center=dict(x=0, y=0, z=0),
    eye=dict(x=0.1, y=2.5, z=0.1)
))

fig.show()
```



e plots

```
import numpy as np
import plotly.graph_objects as go

X, Y, Z = np.mgrid[:1:20j, :1:20j, :1:20j]
vol = (X - 1)**2 + (Y - 1)**2 + Z**2


fig = go.Figure(data=go.Volume(
    x=X.flatten(), y=Y.flatten(), z=Z.flatten(),
    value=vol.flatten(),
    isomin=0.2,
    isomax=0.7,
    opacity=0.2,
    surface_count=21,
    caps= dict(x_show=False, y_show=False, z_show=False), # no caps
    ))
fig.update_layout(scene_camera = dict(
    up=dict(x=0, y=0, z=1),
    center=dict(x=0, y=0, z=0),
    eye=dict(x=0.1, y=2.5, z=0.1)
))

fig.show()
```

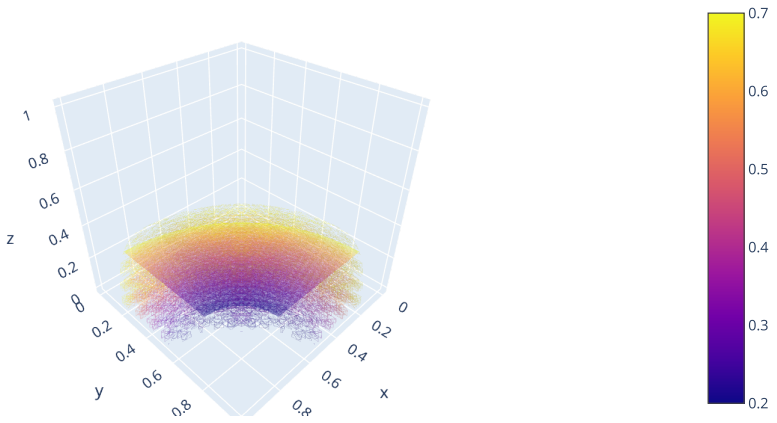


Adding slices to a volume plot

Slices through the volume can be added to the volume plot. In this example the isosurfaces are only partially filled so that the slice is more visible, and the caps were removed for the same purpose.

e plots

```
import numpy as np
import plotly.graph_objects as go

X, Y, Z = np.mgrid[:1:20j, :1:20j, :1:20j]
vol = (X - 1)**2 + (Y - 1)**2 + Z**2


fig = go.Figure(data=go.Volume(
    x=X.flatten(), y=Y.flatten(), z=Z.flatten(),
    value=vol.flatten(),
    isomin=0.2,
    isomax=0.7,
    opacity=0.2,
    surface_count=21,
    slices_z=dict(show=True, locations=[0.4]),
    surface=dict(fill=0.5, pattern='odd'),
    caps= dict(x_show=False, y_show=False, z_show=False), # no caps
    ))

fig.show()
```



Reference

See https://plotly.com/python/reference/volume/ (https://plotly.com/python/reference/volume/) for more information and chart attribute options!

See also

3D isosurface documentation (/python/3d-isosurface-plots/)

What About Dash?

Dash (https://dash.plot.ly/) is an open-source framework for building analytical applications, with no Javascript required, and it is tightly integrated with the Plotly graphing library.

Learn about how to install Dash at https://dash.plot.ly/installation (https://dash.plot.ly/installation).

Everywhere in this page that you see fig.show(), you can display the same figure in a Dash application by passing it to the figure argument of the Graph component (https://dash.plot.ly/dash-core-components/graph) from the built-in dash_core_components package like this:

e plots

```
import plotly.graph_objects as go # or plotly.express as px
fig = go.Figure() # or any Plotly Express function e.g. px.bar(...)
# fig.add_trace( ... )
# fig.update_layout( ... )

from dash import Dash, dcc, html

app = Dash()
app.layout = html.Div([
    dcc.Graph(figure=fig)
])

app.run(debug=True, use_reloader=False)  # Turn off reloader if inside Jupyter
```