Orient-STL: MATH4630 Project

Selecting the best orientation of a print should improve print quality regardless of the FFF printer and slicing software used. Finding it is an optimization problem to minimize overhang surface area by rotating the part.
In this Mathematica Notebook, I verify that my implementation of the orientation objective function is correct.

3D Rotation

Rotation Matrix

```
rx[Tx_] := {{1, 0, 0}, {0, Cos[Tx], -Sin[Tx]}, {0, Sin[Tx], Cos[Tx]}};
ry[Ty_] := {{Cos[Ty], 0, Sin[Ty]}, {0, 1, 0}, {-Sin[Ty], 0, Cos[Ty]}};
rz[Tz_] := {{Cos[Tz], -Sin[Tz], 0}, {Sin[Tz], Cos[Tz], 0}, {0, 0, 1}};
r[T_] := rx[T[[1]]].ry[T[[2]]].rz[T[[3]]];
r[{θx, θy, θz}] // TraditionalForm
```

$$\begin{pmatrix} \cos(\theta_y)\cos(\theta_z) & -\cos(\theta_y)\sin(\theta_z) & \sin(\theta_y) \\ \cos(\theta_z)\sin(\theta_x)\sin(\theta_y)+\cos(\theta_x)\sin(\theta_z) & \cos(\theta_x)\cos(\theta_z)-\sin(\theta_x)\sin(\theta_y)\sin(\theta_z) & -\cos(\theta_y)\sin(\theta_x) \\ \sin(\theta_x)\sin(\theta_z)-\cos(\theta_x)\cos(\theta_z)\sin(\theta_y) & \cos(\theta_z)\sin(\theta_x)+\cos(\theta_x)\sin(\theta_y)\sin(\theta_z) & \cos(\theta_x)\cos(\theta_y) \end{pmatrix}$$

Rotate a Point

```
v1prime[v_, T_] := v.r[T];
FullSimplify[v1prime[{vx, vy, vz}, {θx, θy, θz}], Reals] // MatrixForm //
 TraditionalForm
```

$$\begin{pmatrix} \cos(\theta_z)\,(\mathrm{vx}\cos(\theta_y)+\sin(\theta_y)\,(\mathrm{vy}\sin(\theta_x)-\mathrm{vz}\cos(\theta_x)))+\sin(\theta_z)\,(\mathrm{vy}\cos(\theta_x)+\mathrm{vz}\sin(\theta_x)) \\ \cos(\theta_z)\,(\mathrm{vy}\cos(\theta_x)+\mathrm{vz}\sin(\theta_x))-\sin(\theta_z)\,(\mathrm{vx}\cos(\theta_y)+\sin(\theta_y)\,(\mathrm{vy}\sin(\theta_x)-\mathrm{vz}\cos(\theta_x))) \\ \mathrm{vx}\sin(\theta_y)+\cos(\theta_y)\,(\mathrm{vz}\cos(\theta_x)-\mathrm{vy}\sin(\theta_x)) \end{pmatrix}$$

Rotate a List of 3 Points (Triangle)

```
v3prime[V_, T_] := {v1prime[V[[1]], T], v1prime[V[[2]], T], v1prime[V[[3]], T]};
v3prime[{{V1x, V1y, V1z}, {V2x, V2y, V2z}, {V3x, V3y, V3z}}, {θx, θy, θz}] //
 TraditionalForm
```

$$\left(\begin{matrix} \mathrm{V1x}\cos(\theta_y)\cos(\theta_z)+\mathrm{V1y}\,(\cos(\theta_z)\sin(\theta_x)\sin(\theta_y)+\cos(\theta_x)\sin(\theta_z))+\mathrm{V1z}\,(\sin(\theta_x)\sin(\theta_z)-\cos(\theta_x)\cos(\theta_z)\sin(\theta_y)) & -\mathrm{V} \\ \mathrm{V2x}\cos(\theta_y)\cos(\theta_z)+\mathrm{V2y}\,(\cos(\theta_z)\sin(\theta_x)\sin(\theta_y)+\cos(\theta_x)\sin(\theta_z))+\mathrm{V2z}\,(\sin(\theta_x)\sin(\theta_z)-\cos(\theta_x)\cos(\theta_z)\sin(\theta_y)) & -\mathrm{V} \\ \mathrm{V3x}\cos(\theta_y)\cos(\theta_z)+\mathrm{V3y}\,(\cos(\theta_z)\sin(\theta_x)\sin(\theta_y)+\cos(\theta_x)\sin(\theta_z))+\mathrm{V3z}\,(\sin(\theta_x)\sin(\theta_z)-\cos(\theta_x)\cos(\theta_z)\sin(\theta_y)) & -\mathrm{V} \end{matrix} \right.$$

Objective Function

Define the side vectors (a, b, c) and overhang bias

```
a[V_] := V[[2]] - V[[1]];
b[V_] := V[[3]] - V[[1]];
c[V_] := V[[3]] - V[[2]];
overhangbias[V_] := Tanh[-(a[V][[1]] * b[V][[2]] - a[V][[2]] * b[V][[1]])] / 2 + .5;
overhangbias[{{V1x, V1y, V1z}, {V2x, V2y, V2z}, {V3x, V3y, V3z}}]
```

$$0.5 + \frac{1}{2}\,\mathrm{Tanh}\left[(-V1y + V2y)\ (-V1x + V3x) - (-V1x + V2x)\ (-V1y + V3y)\right]$$

Define the objective function itself (with some substitutions to make it simpler)

```
f[V_, T_] := Module[
   {vprime, aprime, bprime, cprime},
   (* Rotated triangle vertices *)
   vprime = FullSimplify[v3prime[V, T]];
   aprime = a[vprime][[ ;; 2]];
   bprime = b[vprime][[ ;; 2]];
   cprime = c[vprime][[ ;; 2]];
   (* 2-D side lengths of the triangle *)
   aprime = Sqrt[aprime.aprime];
   bprime = Sqrt[bprime.bprime];
   cprime = Sqrt[cprime.cprime];
   (* Multiply overhang bias by triangle area (Heron's formula) *)
   FullSimplify[
    overhangbias[vprime] ×
     Sqrt[(aprime + (bprime + cprime)) (cprime - (aprime - bprime))
        (cprime + (aprime - bprime)) (aprime + (bprime - cprime))] / 4]
  ];
f[{{V1x, V1y, V1z}, {V2x, V2y, V2z}, {V3x, V3y, V3z}}, {θx, θy, θz}]
```

$$0.25 \sqrt{\left(\left(-V1x\ V2y + V1y\ (V2x - V3x) + V2y\ V3x + V1x\ V3y - V2x\ V3y\right) \mathrm{Cos}[\theta_x]\ \mathrm{Cos}[\theta_y] + \left(V1z\ V2x - V1x\ V2z - V1z\ V3x + V2z\ V3x + V1x\ V3z - V2x\ V3z\right) \mathrm{Cos}[\theta_y]\ \mathrm{Sin}[\theta_x] + \left(V1z\ V2y - V1y\ V2z - V1z\ V3y + V2z\ V3y + V1y\ V3z - V2y\ V3z\right) \mathrm{Sin}[\theta_y]\right)^2}$$
$$\left(1. + \mathrm{Tanh}\left[\left(V1y\ V2x - V1x\ V2y - V1y\ V3x + V2y\ V3x + V1x\ V3y - V2x\ V3y\right) \mathrm{Cos}[\theta_x]\ \mathrm{Cos}[\theta_y] + \left(V1z\ V2x - V1x\ V2z - V1z\ V3x + V2z\ V3x + V1x\ V3z - V2x\ V3z\right) \mathrm{Cos}[\theta_y]\ \mathrm{Sin}[\theta_x] + \left(V1z\ V2y - V1y\ V2z - V1z\ V3y + V2z\ V3y + V1y\ V3z - V2y\ V3z\right) \mathrm{Sin}[\theta_y]\right]\right)$$

This looks very complex, but with a few substitutions of the sums, it’s much easier to understand.

```
f[{{V1x, V1y, V1z}, {V2x, V2y, V2z}, {V3x, V3y, V3z}}, {θx, θy, θz}] /.
  {-V1x V2y + V1y (V2x - V3x) + V2y V3x + V1x V3y - V2x V3y → sa,
   V1z V2x - V1x V2z - V1z V3x + V2z V3x + V1x V3z - V2x V3z → sb,
   V1z V2y - V1y V2z - V1z V3y + V2z V3y + V1y V3z - V2y V3z → sc,
   V1y V2x - V1x V2y - V1y V3x + V2y V3x + V1x V3y - V2x V3y → sd
  }
```

$$0.25 \sqrt{\left(\text{sa} \operatorname{Cos}[\theta_x] \operatorname{Cos}[\theta_y] + \text{sb} \operatorname{Cos}[\theta_y] \operatorname{Sin}[\theta_x] + \text{sc} \operatorname{Sin}[\theta_y]\right)^2} \left(1. + \operatorname{Tanh}[\text{sd} \operatorname{Cos}[\theta_x] \operatorname{Cos}[\theta_y] + \text{sb} \operatorname{Cos}[\theta_y] \operatorname{Sin}[\theta_x] + \text{sc} \operatorname{Sin}[\theta_y]]\right)$$

```
Expand[{-V1x V2y + V1y (V2x - V3x) + V2y V3x + V1x V3y - V2x V3y → sa,
   V1z V2x - V1x V2z - V1z V3x + V2z V3x + V1x V3z - V2x V3z → sb,
   V1z V2y - V1y V2z - V1z V3y + V2z V3y + V1y V3z - V2y V3z → sc,
   V1y V2x - V1x V2y - V1y V3x + V2y V3x + V1x V3y - V2x V3y → sd
  }]
```

```
{V1y V2x - V1x V2y - V1y V3x + V2y V3x + V1x V3y - V2x V3y → sa,
 V1z V2x - V1x V2z - V1z V3x + V2z V3x + V1x V3z - V2x V3z → sb,
 V1z V2y - V1y V2z - V1z V3y + V2z V3y + V1y V3z - V2y V3z → sc,
 V1y V2x - V1x V2y - V1y V3x + V2y V3x + V1x V3y - V2x V3y → sd}
```

It turns out that sum a is actually equivalent to sum d, as verified below, so we can substitute sum d for sum a.

```
Expand[sa] == Expand[sd] /. {sa → V1y V2x - V1x V2y - V1y V3x + V2y V3x + V1x V3y - V2x V3y,
   sd → V1y V2x - V1x V2y - V1y V3x + V2y V3x + V1x V3y - V2x V3y}
```

```
True
```

Now, using those substitutions, another substitution arises.

```
f[{{V1x, V1y, V1z}, {V2x, V2y, V2z}, {V3x, V3y, V3z}}, {θx, θy, θz}] /.
   {-V1x V2y + V1y (V2x - V3x) + V2y V3x + V1x V3y - V2x V3y → sa,
    V1z V2x - V1x V2z - V1z V3x + V2z V3x + V1x V3z - V2x V3z → sb,
    V1z V2y - V1y V2z - V1z V3y + V2z V3y + V1y V3z - V2y V3z → sc,
    V1y V2x - V1x V2y - V1y V3x + V2y V3x + V1x V3y - V2x V3y → sd
   } /. {sd → sa}
f[{{V1x, V1y, V1z}, {V2x, V2y, V2z}, {V3x, V3y, V3z}}, {θx, θy, θz}] /.
    {-V1x V2y + V1y (V2x - V3x) + V2y V3x + V1x V3y - V2x V3y → sa,
     V1z V2x - V1x V2z - V1z V3x + V2z V3x + V1x V3z - V2x V3z → sb,
     V1z V2y - V1y V2z - V1z V3y + V2z V3y + V1y V3z - V2y V3z → sc,
     V1y V2x - V1x V2y - V1y V3x + V2y V3x + V1x V3y - V2x V3y → sd
    } /. {sd → sa} /. {sa Cos[θx] Cos[θy] + sb Cos[θy] Sin[θx] + sc Sin[θy] → S}
```

$$0.25 \sqrt{\left(\text{sa} \operatorname{Cos}[\theta_x] \operatorname{Cos}[\theta_y] + \text{sb} \operatorname{Cos}[\theta_y] \operatorname{Sin}[\theta_x] + \text{sc} \operatorname{Sin}[\theta_y]\right)^2} \left(1. + \operatorname{Tanh}[\text{sa} \operatorname{Cos}[\theta_x] \operatorname{Cos}[\theta_y] + \text{sb} \operatorname{Cos}[\theta_y] \operatorname{Sin}[\theta_x] + \text{sc} \operatorname{Sin}[\theta_y]]\right)$$

Now that I've broken down the function, I update my Python implementation to precompute sa, sb, and sc. Note that this increases the amount of error, since I no longer use the numerically

stable Heron's formula. The function remained stable for all the shapes I tried, but it may affect parts with extremely small facets, i.e. fractal shapes like the Menger sponge.

$$0.25\sqrt{S^2}\,(1. + \text{Tanh}[S])$$

It is also important to note that the function above does not include the bottom face bias. This bias depends on the rotation of *all* triangles of a part to find the minimum z-height, which means the function would have to be re-defined in terms of a list of triangles, which is out of the scope of this notebook.

Compute the Jacobian of the Objective Function (with the above substitutions)

```
f'[{{V1x_, V1y_, V1z_}, {V2x_, V2y_, V2z_}, {V3x_, V3y_, V3z_}}, {Tx_, Ty_, Tz_}] = D[f[
    {{V1x, V1y, V1z}, {V2x, V2y, V2z}, {V3x, V3y, V3z}}, {Tx, Ty, Tz}], {{Tx, Ty, Tz}}];
```

```
FullSimplify[D[f[{{V1x, V1y, V1z}, {V2x, V2y, V2z}, {V3x, V3y, V3z}}, {θx, θy, θz}],
     {{θx, θy, θz}}] /.
    {-V1x V2y + V1y (V2x - V3x) + V2y V3x + V1x V3y - V2x V3y → sa,
     V1z V2x - V1x V2z - V1z V3x + V2z V3x + V1x V3z - V2x V3z → sb,
     V1z V2y - V1y V2z - V1z V3y + V2z V3y + V1y V3z - V2y V3z → sc,
     V1y V2x - V1x V2y - V1y V3x + V2y V3x + V1x V3y - V2x V3y → sd
    } /. {sd → sa} /. {sa Cos[θx] Cos[θy] + sb Cos[θy] Sin[θx] + sc Sin[θy] → S}, Reals]
```

$$\left\{\frac{1}{\sqrt{S^2}} S\,\text{Cos}[\theta_y]\,\big(0.25\,\text{sb}\,\text{Cos}[\theta_x] - 0.25\,\text{sa}\,\text{Sin}[\theta_x] + (0.25\,\text{sb}\,\text{Cos}[\theta_x] - 0.25\,\text{sa}\,\text{Sin}[\theta_x])\,(S\,\text{Sech}[S]^2 + \text{Tanh}[S])\big),\right.$$
$$\left.\frac{1}{\sqrt{S^2}} 0.25\,S\,\big(\text{sc}\,\text{Cos}[\theta_y] + (-1.\,\text{sa}\,\text{Cos}[\theta_x] - 1.\,\text{sb}\,\text{Sin}[\theta_x])\,\text{Sin}[\theta_y]\big)\,(1. + 1.\,S\,\text{Sech}[S]^2 + 1.\,\text{Tanh}[S]),\ 0\right\}$$

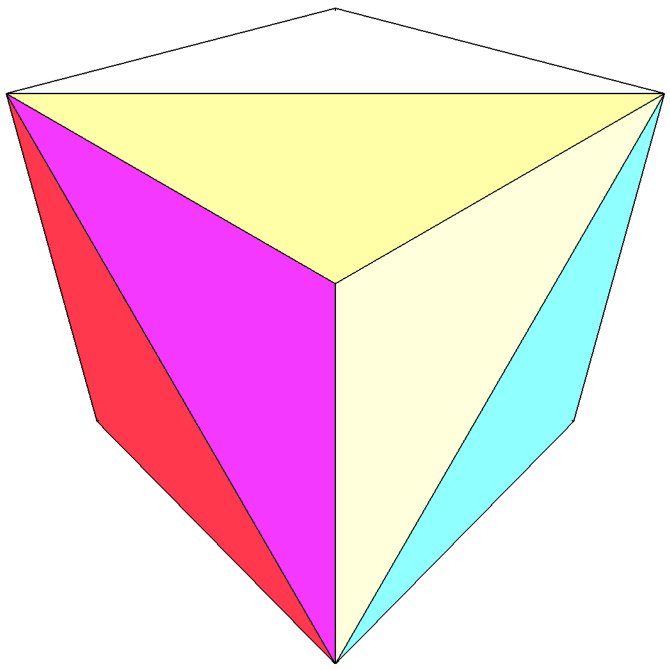
The derivative above also does not match the actual derivative due to the use of the bottom face bias. Even if I included the bottom face bias in the objective function, the derivative would end up being piecewise due to the use of a minimize function in my Python code and be complex to compute. For this reason, I focus on non-derivative methods for optimizing the objective function.

Test the Objective Function

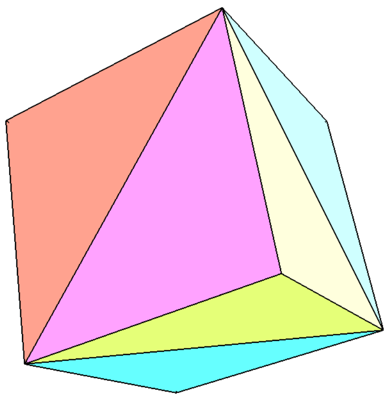
```
stl = Import[
    "~/Documents/projects/orient-stl/examples/cube.stl", "PolygonObjects"][[1]];
rotation = {45 Degree, 45 Degree, 0 Degree};
overhangbias /@ stl
Total[Table[f[stl[[i]], rotation], {i, Length[stl]}] // N]
Total[Table[f'[stl[[i]], rotation], {i, Length[stl]}] // N]
Graphics3D[Table[{Glow@Hue[i / 4], Polygon[stl[[i]]]}, {i, Length[stl]}],
 Boxed → False, ViewCenter → {0, 0, 0}, ViewPoint → {1, 1, 1}]
Graphics3D[Table[{Glow@Hue[i / 4], Polygon[v3prime[stl[[i]], rotation]]},
  {i, Length[stl]}], Boxed → False, ViewCenter → {0, 0, 0}, ViewPoint → {1, 1, 1}]
Plot3D[Total[Table[f[stl[[i]], {x, y, 0}], {i, Length[stl]}] // N],
 {x, -Pi, Pi}, {y, -Pi, Pi}, PlotPoints → 20]
Plot3D[Total[Table[f'[stl[[i]], {x, y, 0}][[1]], {i, Length[stl]}] // N],
 {x, -Pi, Pi}, {y, -Pi, Pi}, PlotPoints → 20]
Plot3D[Total[Table[f'[stl[[i]], {x, y, 0}][[2]], {i, Length[stl]}] // N],
 {x, -Pi, Pi}, {y, -Pi, Pi}, PlotPoints → 20]
```

```
{0.119203, 0.119203, 0.880797, 0.880797, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5}

1.70711

{0., -0.292893, 0.}
```

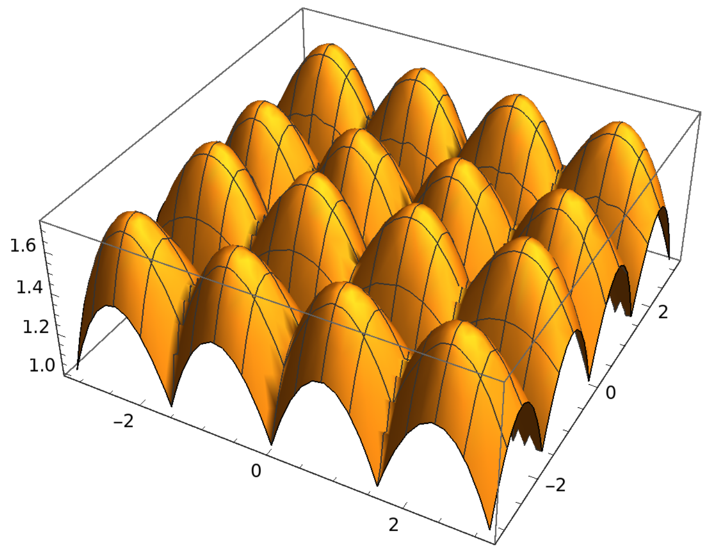
Cube, Flat



Cube, on a Corner

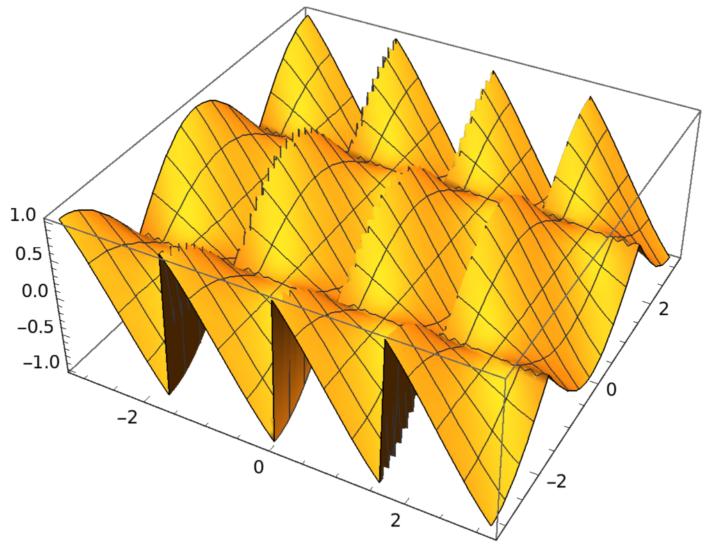


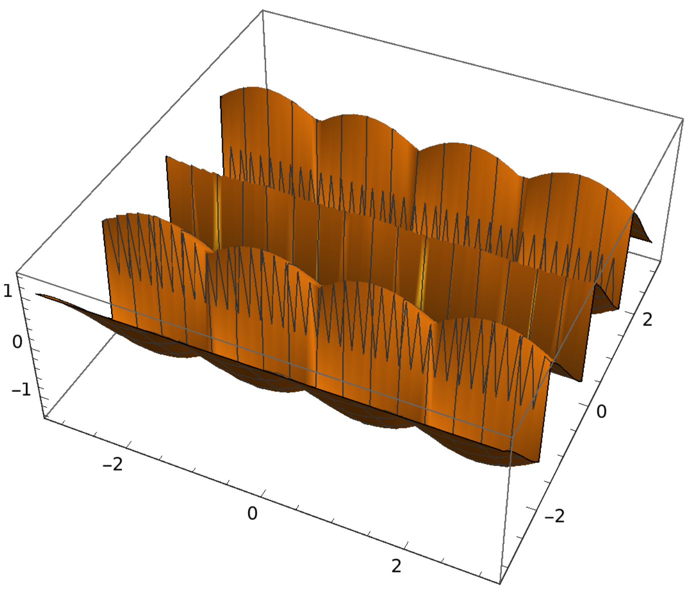
Objective Function Graph



Objective Function Derivative Graphs (dfdTx, dfdTy)

The derivative graphs are interesting and periodic. They are close to what the Python implementation's derivative should look like, but sharp changes created by the bottom face bias are not present. Using this derivative in my Python implementation might mislead optimization methods.




1
0
–1
–2
0
2
2
0
–2