Mesh :: Infinite sheet of cells

generating mesh with periodic boundary condition

In[ ]:=
```
(* subdividing XY grid *)
```

In[ ]:=
```
xL = 13.20; yL = 15.20; cellNumX = 20;
```

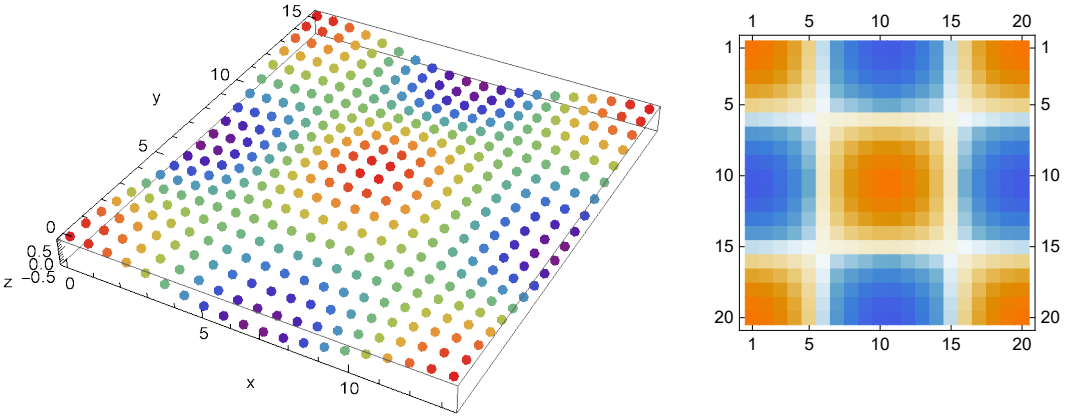
In[ ]:=
```
{δx, δy} = Part[Subdivide[0, #, cellNumX - 1], 2] & /@ {xL, yL};
```

In[ ]:=
```
{x, y, z} = Transpose@Flatten[
    Table[{x, y, 0.5 * Cos[2 π x / xL] Cos[2 π y / yL]}, {x, 0, xL, δx}, {y, 0, yL, δy}], 1];
threadedPts = Thread[{x, y, z}];
cols = ColorData["Rainbow"] /@ Rescale[z];
Print[Style["# of cells in the grid : ", Blue, Bold], Length@threadedPts];
```

```
# of cells in the grid : 400
```

In[ ]:=
```
Grid[{{Graphics3D[{PointSize[0.017], Thread[{cols, Point /@ threadedPts}]},
    Axes → True, AxesLabel → {"x", "y", "z"}, ImageSize → Medium],
   MatrixPlot[Partition[z, cellNumX], ImageSize → Small]}}]
(* grid pts and z coordinate value in the grid represented as matrixplot *)
```

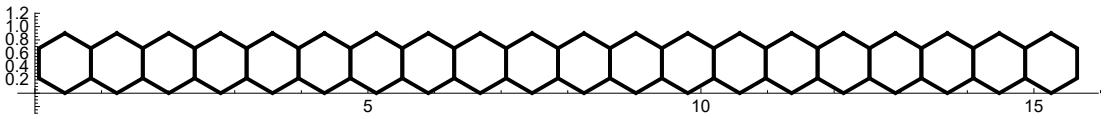
Out[ ]=



```
In[•]:= AllTrue[Through[{First, Last}[#]] & /@ Partition[z, cellNumX], SameQ]
        AllTrue[Through[{First, Last}[#]] & /@ (Partition[z, cellNumX])^T, SameQ]
        (* Z coordinates at the edges are the same along X & Y Axes *)
Out[•]= True

Out[•]= True
```

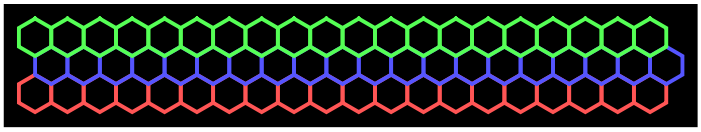
```
In[•]:= (* initial seed for a hexagon face *)
        circlepts = 0.45 CirclePoints[{1, 1}, {1, 0}, 6];
        pts = Reverse[NestList[{0, 0.45 Sqrt[3]} + # & /@ # &, circlepts, cellNumX - 1], {3}];

In[•]:= Graphics[{Black, Thick, Line[# ~Append~ First[#]] & /@ pts}, Axes → True, ImageSize → Large]
```

Out[•]=




```
In[•]:= Block[{col2, col3},
          col2 = Map[0.45 {Sqrt[3] / 2, 3 / 2} + # &, pts, {2}];
          col3 = Map[0.45 {-Sqrt[3] / 2, 3 / 2} + # &, col2, {2}];
          Show[{Graphics[{Thick, Lighter@Red, Line[# ~Append~ First[#]] & /@ pts}],
            Graphics[{Thick, Lighter@Blue, Line[# ~Append~ First[#]] & /@ col2}],
            Graphics[{Thick, Lighter@Green, Line[# ~Append~ First[#]] & /@ col3}]},
           Background → Black]
        ]
```

Out[•]=

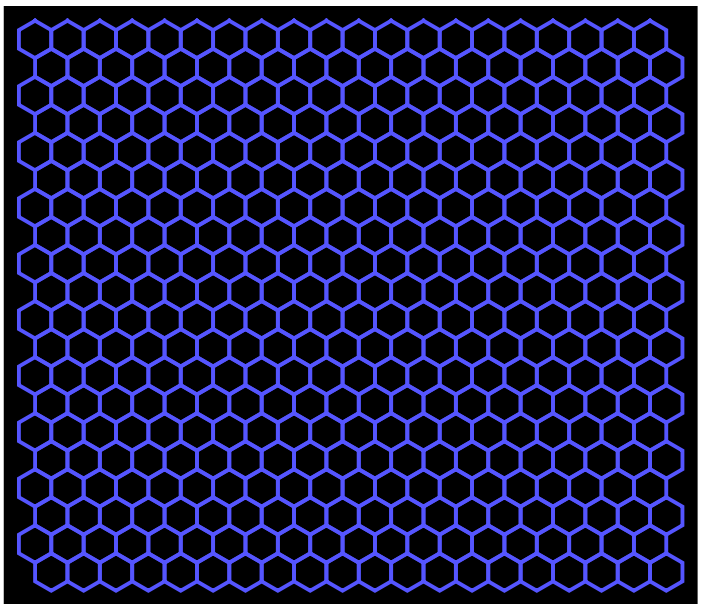


getting a hexagonal lattice

```
In[•]:= (* We generate a 2D hexagonal grid; Add a mirror Z plane;
        Make pairs and Construct Polyhedra *)

In[•]:= grid = Block[{temp, k = 1, p = cellNumX},
            NestList[(temp = Map[x ↦ 0.45 {If[OddQ[k], -1, 1] (Sqrt[3] / 2), 3 / 2} + x, #, {2}];
               k++;
               temp) &, pts, p - 1]
          ];
```

In[]:=
```
Graphics[{Thick, Lighter@Blue, Map[Line[# ~Append~ First[#]] &, grid, {2}]},
 Background → Black, ImageSize → Medium]
```

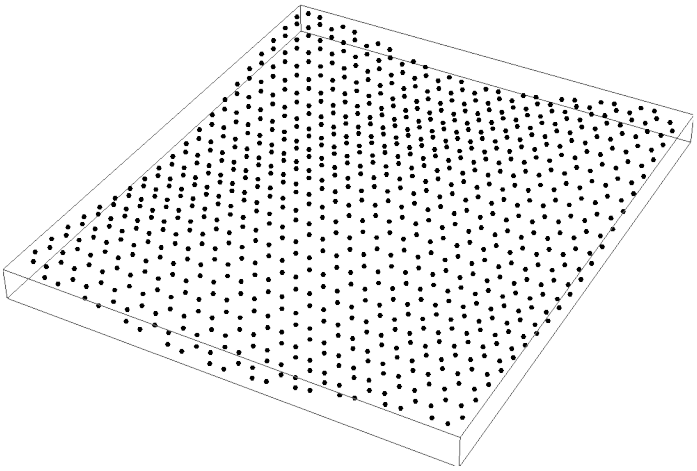
Out[]=



In[]:=
```
xyzPts = Table[{Last[i], First[i], (0.5) * Cos[2 π Last[i] / xL] Cos[2 π First[i] / yL]},
   {i, Flatten[grid, 2]}];
(*adding Z coordinate (height) to the polgonal face *)
```

In[]:=
```
Graphics3D[{Point@DeleteDuplicates@xyzPts}, ImageSize → Medium]
```

Out[]=



In[]:=
```
nf = Nearest[DeleteDuplicates@xyzPts]
```

Out[]=
```
NearestFunction[ ⊞ ⊡ Data points: 1167
                     Input dimension: 3 ]
```

In[]:=
```
(nf[#, {All, 1.0 × 10^-15}] & /@ DeleteDuplicates[xyzPts]) //
 Counts@*Map[Length] (* # of pts for which keys are > 1 need merging because of
    very tiny numerical errors. All this could have been avoided in the first place *)
```

Out[]=
```
⟨|3 → 57, 2 → 498, 1 → 612|⟩
```

In[]:=
```
replacements = Dispatch@Flatten@
   Map[Function[x, # → Mean[x] & /@ x],
    SortBy[Cases[Gather[DeleteDuplicates@xyzPts,
       EuclideanDistance[#1, #2] ≤ 1.0 × 10^-15 &], x_ /; Length[x] > 1],
     First]
    ]
```

Out[]= Dispatch[Length: 555]

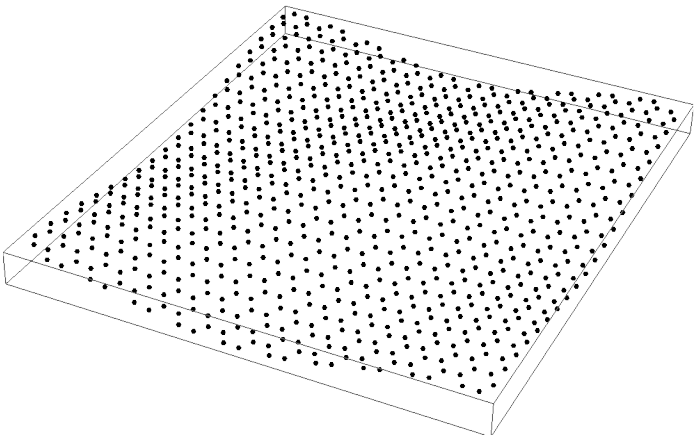
In[]:=
```
(*replacements=Dispatch@Flatten[(x↦First[x]→ #&/@Rest[x])/@Cases[
     Gather[DeleteDuplicates@Flatten[grid,2],EuclideanDistance[#1,#2]≤ 1.0 10^-15&],
     x_/;Length[x]>1],1]*)
```

In[]:=
```
xyzPts = xyzPts /. replacements;
```

In[]:=
```
Print[Length@DeleteDuplicates@xyzPts]
Graphics3D[Point@DeleteDuplicates@xyzPts, ImageSize → Medium]
```

```
880
```

Out[]=



In[]:=
```
nf = Nearest[DeleteDuplicates@xyzPts]
```

Out[]= NearestFunction[Data points: 880 Input dimension: 3]

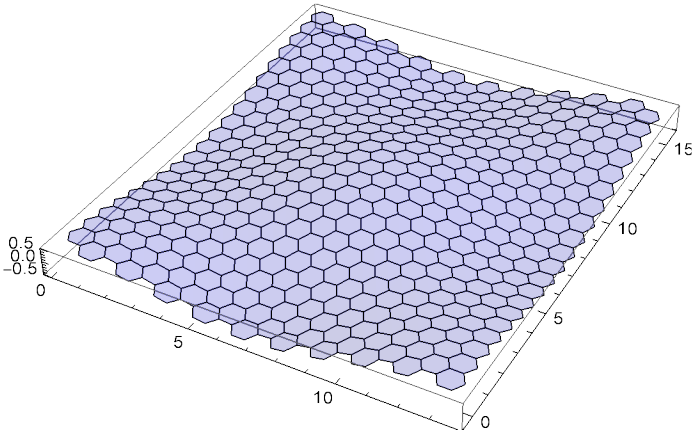
In[]:=
```
Apply[And] //@ (Map[Through[{Apply[SameQ], Apply[Equal]}[#]] &]@Cases[
     nf[#, {All, 1.0 × 10^-15}] & /@ DeleteDuplicates[xyzPts],
     x_ /; Length[x] > 1])
  (*if value is true then the polygons have successfully shared common vertices *)
```

Out[]= True

In[]:=
```
XYZgrid = ArrayReshape[xyzPts, Dimensions[grid] /. {x___, y_} :→ {x, 3}];
```

In[ ]:=
```
Graphics3D[{Opacity[0.2], Blue, Polygon /@ XYZgrid}, Axes → True]
```

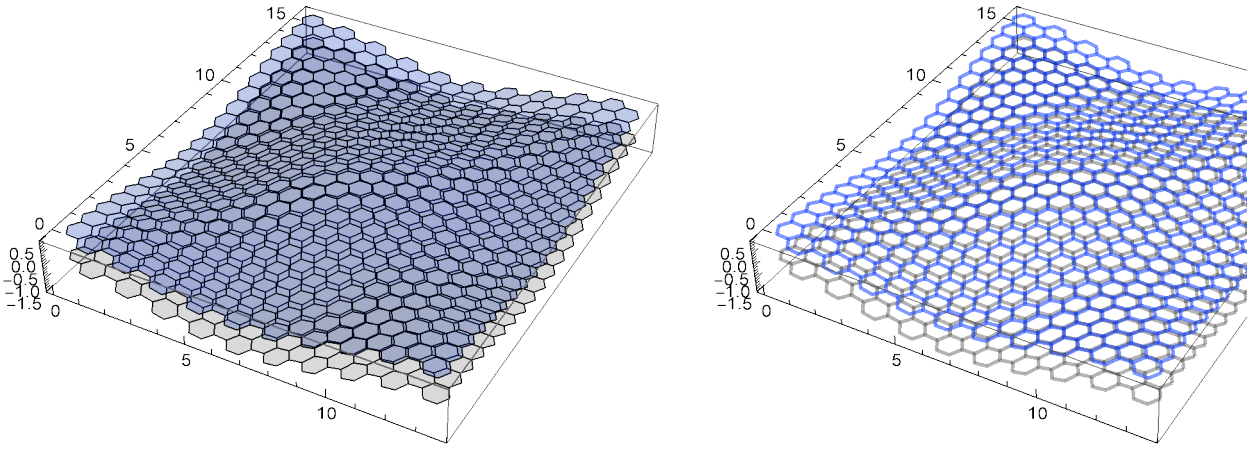
Out[ ]=



In[ ]:=
```
(* Adding a mirror Z plane; putting vertical edges between the
 top and bottom faces and constructing individual polyhedrons or cells;
Later we can add some noise to displace vertices from regularity *)
```

In[ ]:=
```
Grid[{{Graphics3D[{{XYZColor[0, 0, 1, 0.25], Thick, Polygon /@ XYZgrid},
      {XYZColor[0, 0, 0, 0.15], Thick, Polygon /@ Map[# - {0, 0, 1} &, XYZgrid, {3}]}},
     Axes → True, ImageSize → Medium],
    Graphics3D[{{XYZColor[0, 0, 1, 0.5], Thick,
       Map[Line[# ~ Append ~ First[#]] &, XYZgrid, {2}]}, {XYZColor[0, 0, 0, 0.3], Thick,
       Map[Line[# ~ Append ~ First[#]] &, #, {2}] &@Map[# - {0, 0, 1} &, XYZgrid, {3}]}},
     Axes → True, ImageSize → Medium]}}]
```

Out[ ]=



In[ ]:=
```
vertexNum = Length@xyzPts
```

Out[ ]=
```
2400
```

generating the hexagonal prism lattice

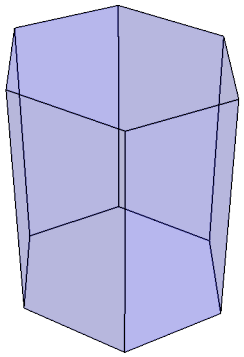
In[ ]:=
```
Clear@triangulateFaces;
triangulateFaces::header =
  "triangulateFaces takes in a list of all the faces of a cell
    and returns a triangulation for each face";
triangulateFaces[faces_] := Block[{edgelen, ls, mean},
   (ls = Partition[#, 2, 1, 1];
      edgelen = EuclideanDistance @@@ ls;
      mean = Total[edgelen * (Midpoint /@ ls)] / Total[edgelen];
      mean  = mean ~ SetPrecision ~ 10;
      Map[Append[#, mean] &, ls]) & /@ faces
  ];
```

In[ ]:=
```
Pts = MapIndexed[{#2[[1]], #1} &,
   Flatten[Riffle[Flatten[XYZgrid, 1], Flatten[Map[# - {0, 0, 1} &, XYZgrid, {3}], 1]], 1]];
 (* all gird pts labeled *)
```

In[ ]:=
```
(*an seed polyhedron*)
seedPolyhedra = Pts[[#]][[All, 2]] & /@ {{1, 2, 3, 4, 5, 6},
    {7, 12, 11, 10, 9, 8}, Sequence @@ NestList[# + 1 &, {1, 7, 8, 2}, 4], {6, 12, 7, 1}};
```

In[ ]:=
```
Graphics3D[{Directive[Blue, Opacity[0.25 * 1 / GoldenRatio]], Polyhedron[seedPolyhedra]},
 Boxed → False, ImageSize → Small]
```

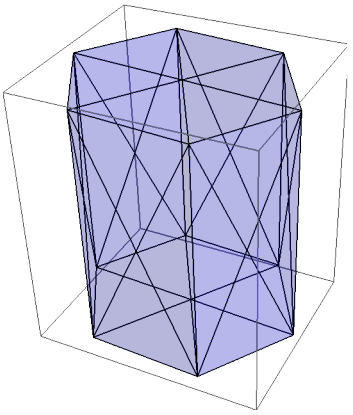
Out[ ]=



In[ ]:=
```
indToPtsAssoc = <|Rule @@@ Pts|>;(* Association of gridptLabel → gridpt *)
```

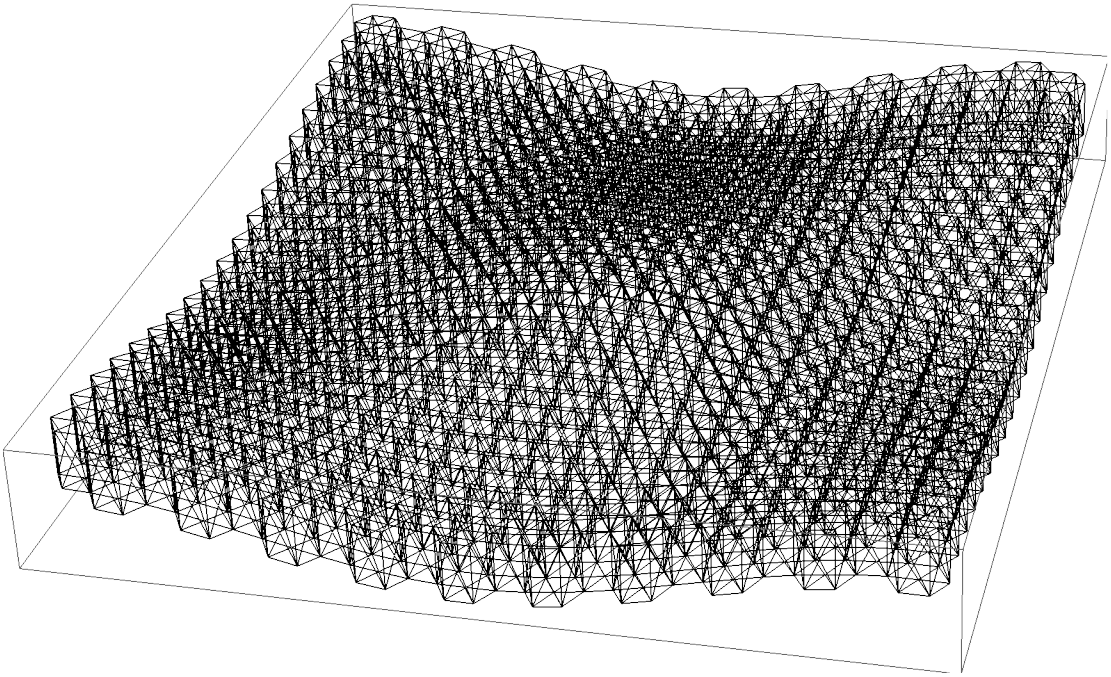
In[•]:=
```
(*we create our polyhedra mesh *)
initFace = {{1, 2, 3, 4, 5, 6}, {7, 12, 11, 10, 9, 8},
   Sequence @@ NestList[# + 1 &, {1, 7, 8, 2}, 4], {6, 12, 7, 1}};
faceListInd = NestList[# + 12 &, initFace, (cellNumX * cellNumX) - 1];
faceListCoords = MapAt[Lookup[indToPtsAssoc, #] &, faceListInd, {All, All, All}];
firstPolyhedra = First@faceListCoords;
Graphics3D[{Directive[Blue, Opacity[0.25 * 1 / GoldenRatio]],
  Polyhedron@Flatten[triangulateFaces[firstPolyhedra], 1]}, ImageSize → Small]
```

Out[•]=



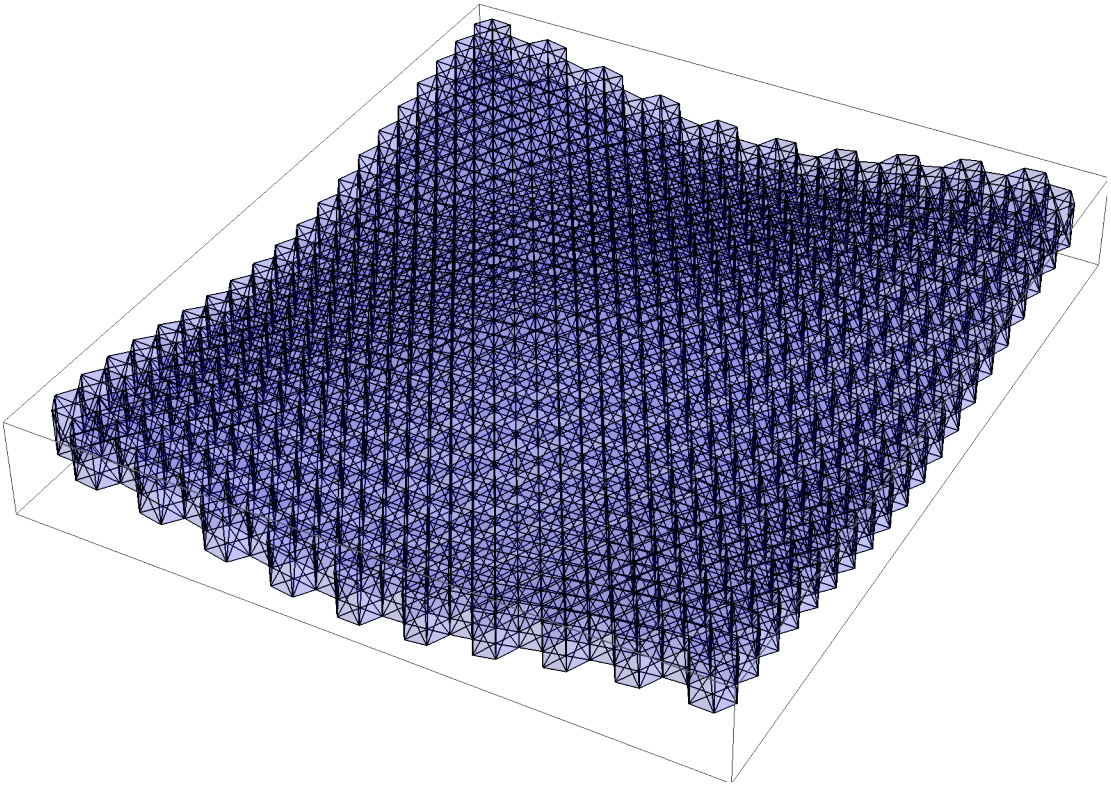
In[•]:=
```
(*wireframe mesh rendering*)
Graphics3D[{Line@Flatten[(triangulateFaces /@ faceListCoords), 2]}, ImageSize → Large]
```

Out[•]=



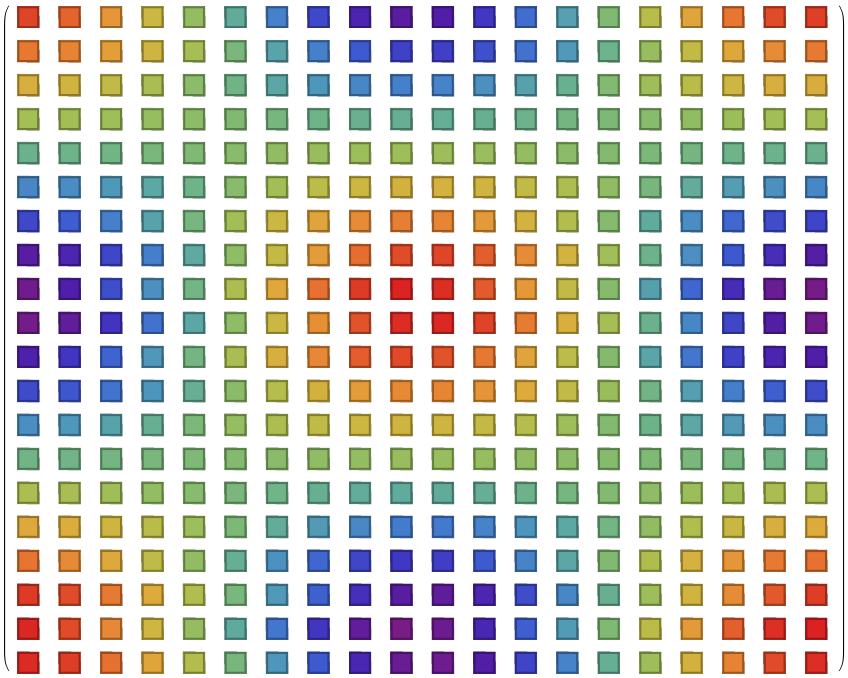
In[•]:=
```
(*polyhedra mesh rendering*)
Graphics3D[{Directive[Blue, Opacity[0.2 * 1 / GoldenRatio]],
  Polyhedron@Flatten[triangulateFaces /@ faceListCoords, 2]}, ImageSize → Large]
```

Out[•]=



```
In[•]:= (col = ColorData["Rainbow"] /@ Rescale[#, {Min[#], Max[#]}, {0, 1}] &[
          Flatten@XYZgrid[[All, All, 1, 3]]]) // Partition[#, cellNumX] & // MatrixForm
```

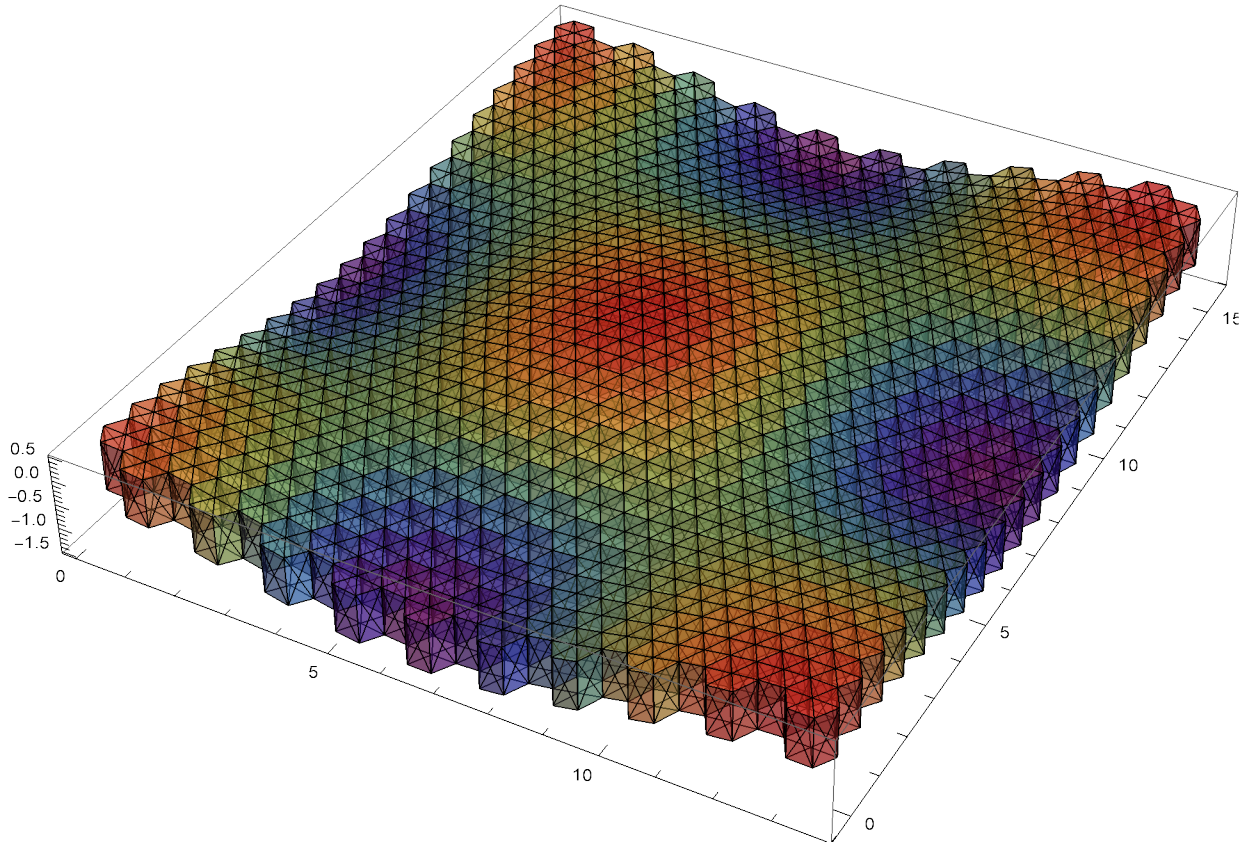
Out[•]//MatrixForm=



```
In[•]:= indToPtsAssoc = AssociationThread[Range[Length@#], #] &[
          DeleteDuplicatesBy[Pts[[All, 2]], N]];(* <| ptlabel → pt .. |> *)
        ptsToIndAssoc = <|Reverse[Normal@indToPtsAssoc, 2]|>;(* <| pt → ptlabel .. |> *)

In[•]:= facePartitions = DeleteMissing[
          Map[Lookup[ptsToIndAssoc, #] &, triangulateFaces /@ faceListCoords, {3}], 4];
        (*polyhedra faces represented as edges via usage of consecutive vertices *)

In[•]:= vertexformingFaces = Map[Lookup[ptsToIndAssoc, #] &, faceListCoords, {2}];
        (* polyhedra represented as sets of vertex labels *)

In[•]:= cellNum = cellNumX^2;

In[•]:= cellVertexGrouping = AssociationThread[Range[cellNum], vertexformingFaces];
        (* <|cell_index → {{vertex labels ..} ..}|> *)

In[•]:= vertexToCell = <|ParallelMap[
            # → Union@Position[facePartitions, #][[All, 1]] &, Range[Max@Keys@indToPtsAssoc]]|>;
        (* <| vertex → {cells the vertex is a member of} .. |> *)

In[•]:= edges = DeleteDuplicatesBy[Flatten[(x ↦ Partition[#, 2, 1, 1] & /@ x) /@ faceListCoords, 2],
          Sort];(*edges in the mesh *)

In[•]:= faceListCoords = Values@Map[indToPtsAssoc, cellVertexGrouping, {3}];
        (*all polyhedra represented by vertex coordinates*)
```

```
In[•]:= Graphics3D[{Directive[Opacity[0.5]],
   Thread[{col, Polyhedron@Flatten[#, 1] & /@ (triangulateFaces /@ faceListCoords)}]},
  ImageSize → Full, Axes → True]
```

Out[•]=




```
In[•]:= (* The mesh above is not stitched about the boundaries;
  we have to stitch the boundaries together *)
```

Stitching Boundaries

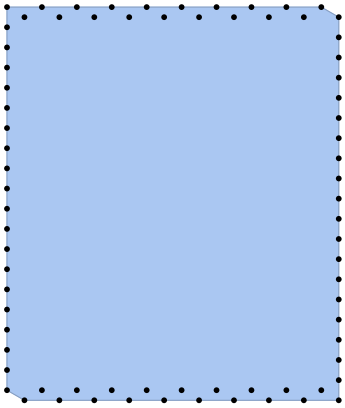
```
In[•]:= (* finding boundary cells *)
```

```
In[•]:= ptsp = # → Mean@Flatten[Map[Most@Lookup[indToPtsAssoc, #] &,
          cellVertexGrouping[#], {2}], 1] & /@ Range[cellNum];
```

```
In[•]:= bcellsH =
    Transpose[Table[{k + 1, cellNumX + k}, {k, cellNumX, cellNumX * (cellNumX - 2), cellNumX}]];
  bcells = Union[Range[cellNumX] ~Join~
      Range[cellNum - cellNumX + 1, cellNum] ~Join~ Flatten[bcellsH]];
```

In[ ]:=
```
Show[{ConvexHullMesh[#], Graphics@Point[#]}, ImageSize → Small] &[
 Part[Values[ptsp], bcells]]
```

Out[ ]=

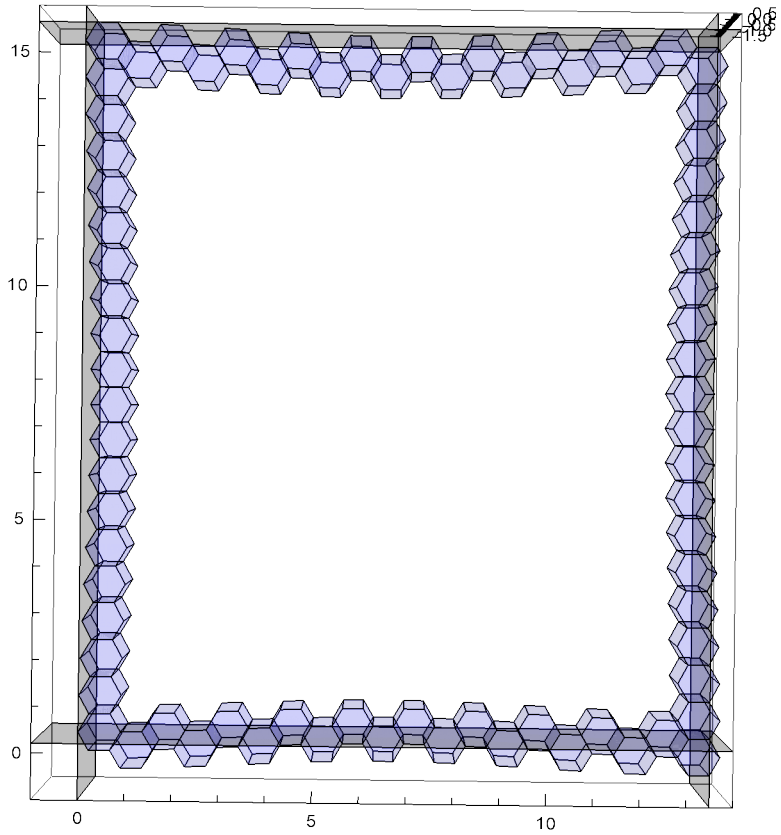


In[ ]:=
```
(* lets specify some limits for the boundaries *)
```

In[ ]:=
```
xLim = {0.`, 13.50`};
yLim = {0.20`, 15.65`};
```

In[•]:=
```
Graphics3D[{
  {Blue, Opacity[0.1],
   Polyhedron@Map[Lookup[indToPtsAssoc, #] &, cellVertexGrouping[#], {2}] & /@
    bcells[[1 ;; 20]]},
  {Black, Opacity[0.25], InfinitePlane[{{xLim[[1]], 0, 0},
     {xLim[[1]], 1, 0}, {xLim[[1]], 1, 1}}]},
  {Blue, Opacity[0.1], Polyhedron@Map[Lookup[indToPtsAssoc, #] &,
       cellVertexGrouping[#], {2}] & /@ bcells[[-20 ;;]]},
  {Black, Opacity[0.25], InfinitePlane[{{xLim[[2]], 0, 0},
     {xLim[[2]], 1, 0}, {xLim[[2]], 1, 1}}]},
  {Blue, Opacity[0.1], Polyhedron@Map[Lookup[indToPtsAssoc, #] &,
       cellVertexGrouping[#], {2}] & /@ First@bcellsH},
  {Black, Opacity[0.25], InfinitePlane[{{0, yLim[[1]], 0},
     {1, yLim[[1]], 0}, {1, yLim[[1]], 1}}]},
  {Blue, Opacity[0.1], Polyhedron@Map[Lookup[indToPtsAssoc, #] &,
       cellVertexGrouping[#], {2}] & /@ Last@bcellsH},
  {Black, Opacity[0.25], InfinitePlane[{{0, yLim[[2]], 0},
     {1, yLim[[2]], 0}, {0, yLim[[2]], 1}}]}
 }, ImageSize → 400, PlotRange → {{-1, 14}, {-1, 16}, {-1.6, 0.5}},
 ViewPoint → Above, Axes → True]
```

Out[•]=



In[•]:=
```
(*We need a transformation matrix for each flagged cell →
 this can be obtained by subtracting pt coordinates from the boundary limits;
storage matrix needs to be created for each flagged cell →
 nearest neighbour function to be used for pts that pass boundary limits;*)
```

In[ ]:=
```
(*we choose two boundaries to create a
 nearest function to stitch the opposite boundaries *)
nearestFunc = Nearest@DeleteDuplicates@Flatten[DeleteDuplicates@Flatten[
        Map[Lookup[indToPtsAssoc, #] &, cellVertexGrouping[#], {2}], 1] & /@
     Join[Range@cellNumX, Last@bcellsH], 1]
```

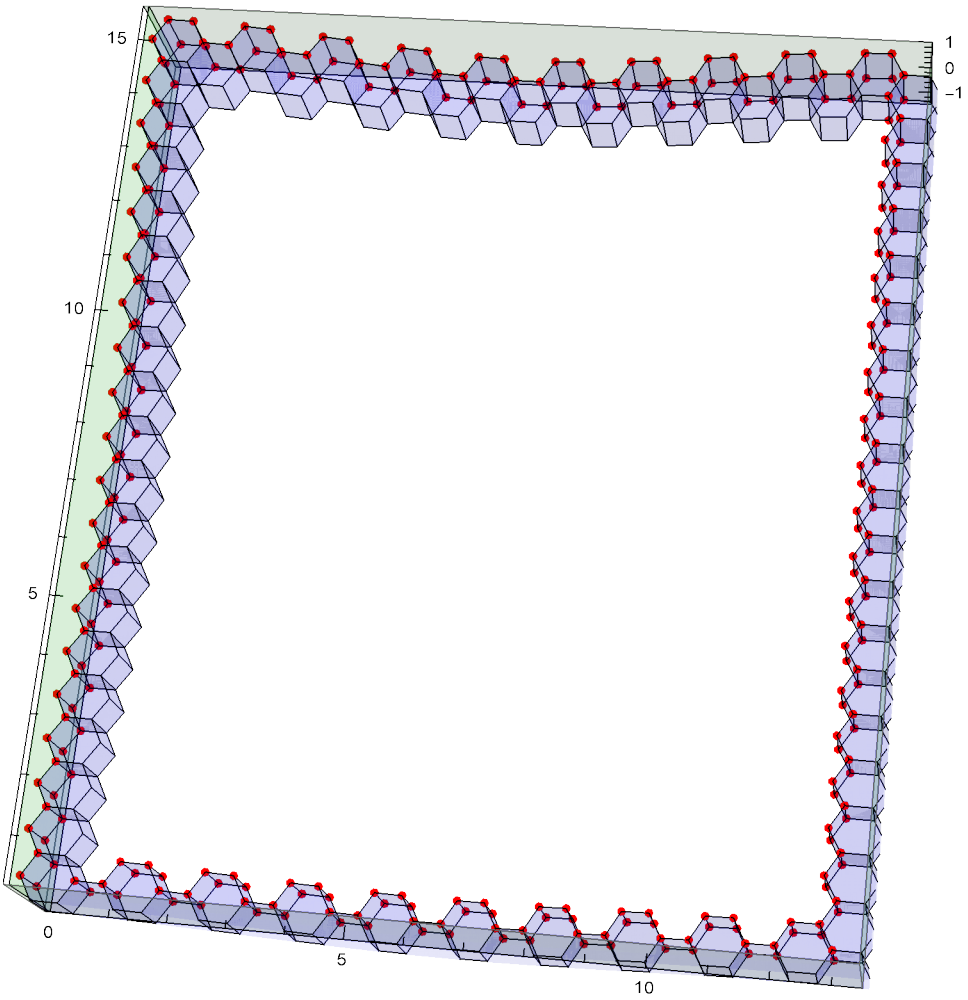
Out[ ]= NearestFunction[Data points: 306 Input dimension: 3]

In[ ]:=
```
With[{xlim2 = xLim[[2]], ylim1 = yLim[[1]], ylim2 = yLim[[2]]},
 periodicRules =
  Dispatch[
   {{x_ /; x ≥ xlim2, y_ /; y ≤ ylim1, z_} :↦ First@nearestFunc[{x - xlim2, y + ylim2, z}],
    {x_ /; x ≥ xlim2, y__} :↦ First@nearestFunc[ {x - xlim2, y}],
    {x_, y_ /; y ≤ ylim1, z_} :↦ First@nearestFunc[{x, y + ylim2, z}]}];

 transformRules = Dispatch[{{x_ /; x ≥ xlim2, y_ /; y ≤ ylim1, _} :↦ {xlim2, -ylim2, 0.},
    {x_ /; x ≥ xlim2, y__} :↦ {xlim2, 0., 0.},
    {x_, y_ /; y ≤ ylim1, z_} :↦ {0., -ylim2, 0.},
    {___Real} :↦ {0., 0., 0.}
   }];
 {periodicRules, transformRules}
]
```

Out[ ]= {Dispatch[Length: 3], Dispatch[Length: 4]}

In[•]:=
```
With[{xlim2 = xLim[[2]], ylim1 = yLim[[1]], ylim2 = yLim[[2]]},
 Graphics3D[
  {Function[ind, {
      {Red, PointSize[0.01], Point /@
        (Map[Lookup[indToPtsAssoc, #] &, cellVertexGrouping[ind], {2}] /. periodicRules)}
     }] /@ Join[First@bcellsH, Range[cellNum - cellNumX + 1, cellNum]],
   {Blue, Opacity[0.1],
    Polyhedron@Map[Lookup[indToPtsAssoc, #] &, cellVertexGrouping[#], {2}] & /@
     Join[Range@cellNumX, Range[cellNum - cellNumX + 1, cellNum],
      First@bcellsH, Last@bcellsH]},
   {Green, Opacity[0.15], InfinitePlane[{{0, 0, 0}, {0, 1, 0}, {0, 1, 1}}],
    InfinitePlane[{{xlim2, 0, 0}, {xlim2, 1, 0}, {xlim2, 1, 1}}],
    InfinitePlane[{{0, ylim1, 0}, {1, ylim1, 0}, {1, ylim1, 1}}],
    InfinitePlane[{{0, ylim2, 0}, {1, ylim2, 0}, {0, ylim2, 1}}]}},
  ImageSize → 500, Axes → True, PlotRange → {{-0.1, xlim2}, {ylim1, ylim2}, {-1.5, 1.0}}
 ]
]
```

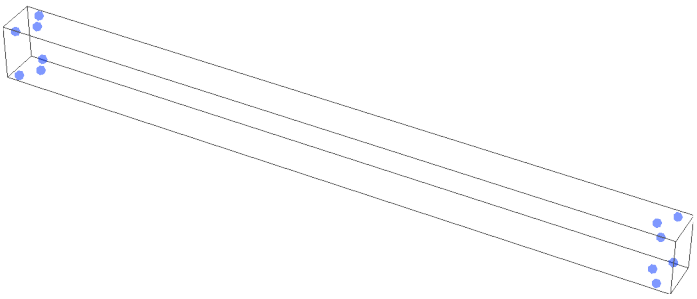
Out[•]=



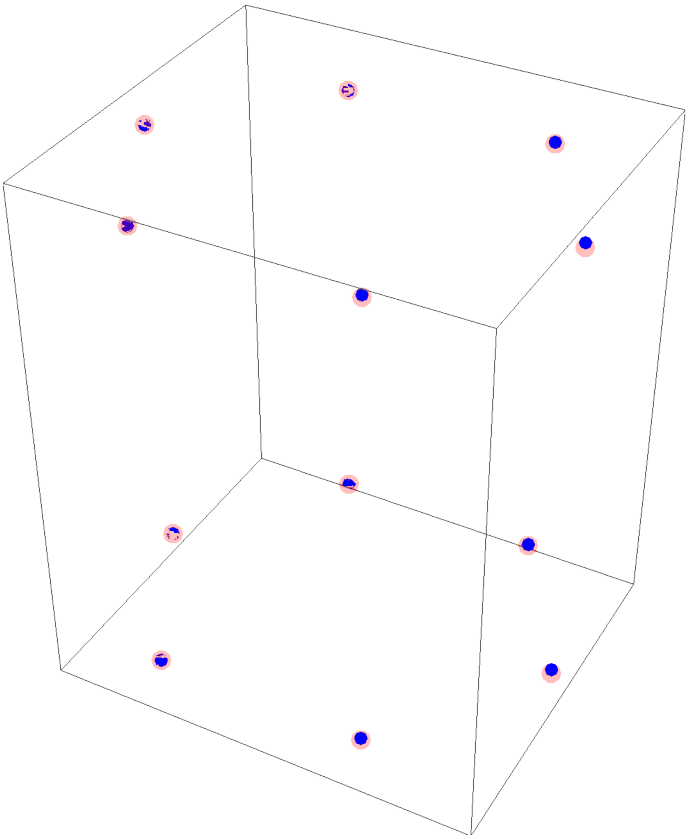
In[•]:=
```
(* here is one boundary cell (#383) wrapping around on the opposite side *)
```

In[•]:=
```
exampleCell = 383;
Block[{cvg = cellVertexGrouping[exampleCell], tMat},
 Print["wrapping coordinates around using nearest neighbours"];
 Print[Map[Lookup[indToPtsAssoc, #] &, cvg, {2}] /. periodicRules //
   Graphics3D[{XYZColor[0, 0, 1, 0.5], PointSize[0.015], #}] &@* Map[Point]];
 tMat = (Map[Lookup[indToPtsAssoc, #] &, cvg, {2}] /. transformRules);
 Print["wrapped coordinates transformed to geometrically correct positions"];
 Print@Graphics3D[{{PointSize[0.030], Opacity[0.25],
     Red, Point /@ Map[Lookup[indToPtsAssoc, #] &, cvg, {2}]},
    {PointSize[0.02], Blue, Point /@
      ((Map[Lookup[indToPtsAssoc, #] &, cvg, {2}] /. periodicRules) + tMat)}},
   ImageSize → Medium];
]
```

wrapping coordinates around using nearest neighbours



wrapped coordinates transformed to geometrically correct positions



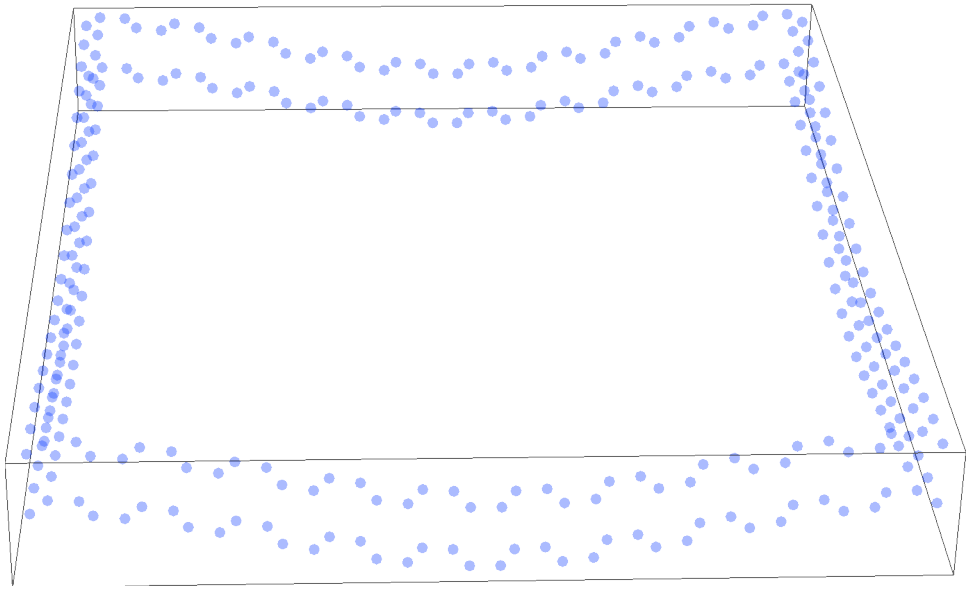
In[ ]:=
```
flaggedCells = {1} ~ Join ~ First@bcellsH ~ Join ~ Range[cellNum - cellNumX + 1, cellNum]
```

Out[ ]=
```
{1, 21, 41, 61, 81, 101, 121, 141, 161, 181, 201, 221, 241, 261, 281, 301, 321, 341, 361, 381,
 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400}
```

In[ ]:=
```
flaggedCellsTransPts =
  Function[ind, ind → (Map[Lookup[indToPtsAssoc, #] &, cellVertexGrouping[ind], {2}] /.
      transformRules)] /@ flaggedCells;
flaggedCellsStoragePts = Function[ind, ind → (Map[Lookup[indToPtsAssoc, #] &,
      cellVertexGrouping[ind], {2}] /. periodicRules)] /@ flaggedCells;
```

In[•]:=
```
Graphics3D[{PointSize[0.012], XYZColor[0, 0, 1, 0.33],
  Map[Point, flaggedCellsStoragePts[[#, 2]] & /@ Range[Length@flaggedCells], {2}]},
 ImageSize → 500]
```

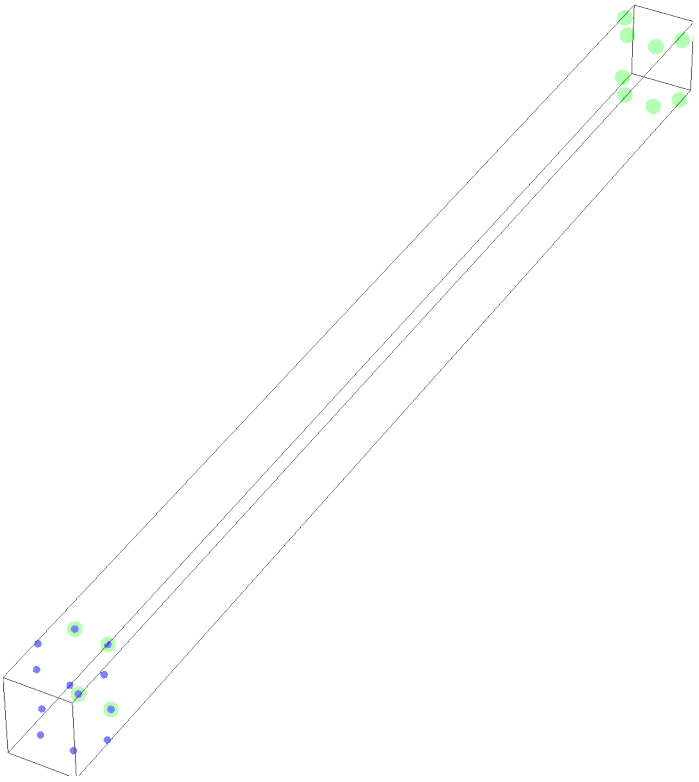
Out[•]=



In[•]:=
```
AssociateTo[cellVertexGrouping,
  MapAt[ptsToIndAssoc, flaggedCellsStoragePts, {All, 2, All, All}]];
```

In[•]:=
```
Block[{exampleCell = 21, pos},
 pos = First @@ Position[flaggedCellsTransPts, exampleCell];
 Graphics3D[{{Green, Opacity[0.3], PointSize[0.024], Point /@ #},
     {Blue, Opacity[0.5], PointSize[0.012],
      Point /@ (# + Values[flaggedCellsTransPts[[pos]]])}},
    ImageSize → Medium] &[Map[Lookup[indToPtsAssoc, #] &,
   cellVertexGrouping[exampleCell], {2}]]
]
```

Out[•]=

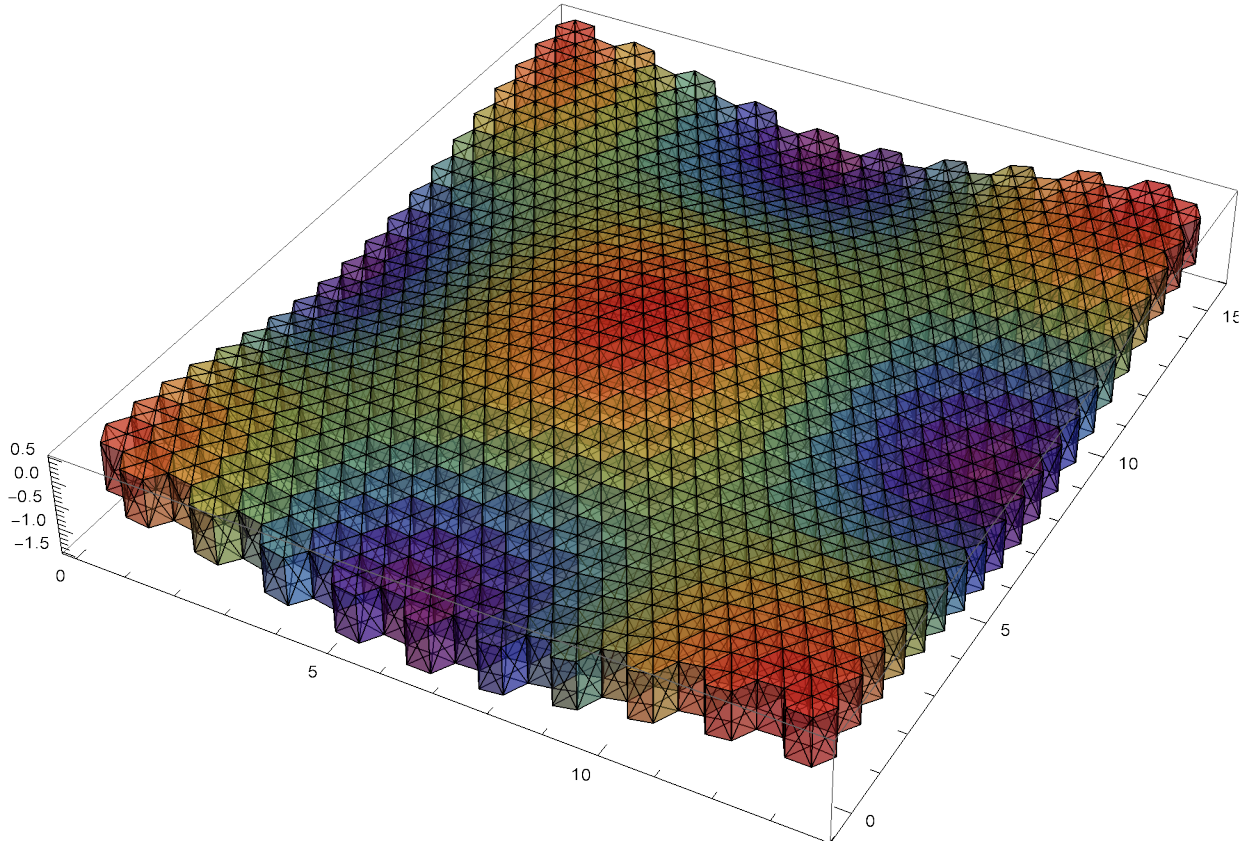


In[•]:=
```
faceListCoords = Map[indToPtsAssoc, cellVertexGrouping, {3}];
```

In[•]:=
```
faceListCoords = Merge[{faceListCoords, <|flaggedCellsTransPts|>}, Total];
faceListCoords = Values@faceListCoords;
```

```
In[•]:= Graphics3D[{Directive[Opacity[0.5]],
   Thread[{col, Polyhedron@Flatten[#, 1] & /@ (triangulateFaces /@ faceListCoords)}]},
  ImageSize → Full, Axes → True]
```

Out[•]=




```
In[•]:= (* the mesh above is complete and is wrapped
   around the boundaries to form an infinite sheet of cells *)

In[•]:= edges = DeleteDuplicatesBy[
    Flatten[(x ↦ Partition[#, 2, 1, 1] & /@ x) /@ faceListCoords, 2], Sort];(* mesh edges *)

In[•]:= indToPtsAssoc =
   AssociationThread[Range[Length@#], #] &[DeleteDuplicates[Flatten[faceListCoords, 2]]];
  (* <| vertex_labels → vertex_pts |> *)

In[•]:= ptsToIndAssoc = <|Reverse[Normal@indToPtsAssoc, 2]|>;
  (* <| vertex_pts → vertex_labels |>*)

In[•]:= cellVertexGrouping = AssociationThread[Range[Length@#], #] &[
    Map[Lookup[ptsToIndAssoc, #] &, faceListCoords, {2}]];
  (* <|cell → {{face vertex labels ..} ..}|>*)

In[•]:= vertexToCell = GroupBy[
    Reverse[
     Flatten[Thread /@ MapAt[Union@* Flatten, Normal[cellVertexGrouping], {All, 2}]], 2],
    First → Last];

In[•]:= (* <| vertex → neighbouring cells .. |> *)
```

```
In[ ]:= (* no duplicate pts found within tiny δ's *)
```

```
In[ ]:= DeleteCases[
  Lookup[ptsToIndAssoc, #] & /@
   With[{vals = Values@indToPtsAssoc},
    With[{nearest = Nearest[vals]},
     nearest[#, {2, 1.0 × 10^-15}] & /@ vals
    ]
   ], x_ /; Length[x] == 1] // Length
```

```
Out[ ]= 0
```

```
In[ ]:= Count[ Lookup[ptsToIndAssoc, #] & /@
  With[{vals = DeleteDuplicates@Flatten[edges, 1]},
   With[{nf = Nearest@vals},
    nf[#, {2, 1.0 × 10^-15}] & /@ vals
   ]
  ], {Repeated[_Integer, {2, ∞}]}]
```

```
Out[ ]= 0
```

```
In[ ]:= Length[
 With[{vals = DeleteDuplicates@Flatten[edges, 1]},
   With[{nf = Nearest@vals},
    nf[#, {2, 1.0 × 10^-15}] & /@ vals
   ]
  ]~DeleteCases~{{___Real}}
]
```

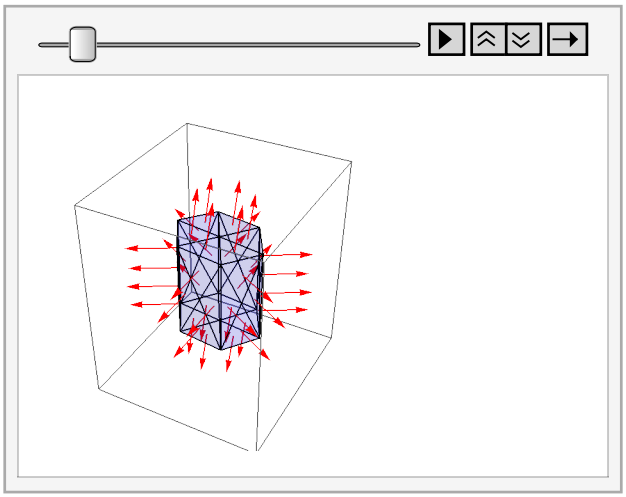
```
Out[ ]= 0
```

check whether normals of triangulations are pointing outwards i.e. faces are oriented c.c.w

```
In[ ]:= plts = Function[ind,
    triangles = Triangle /@ Flatten[triangulateFaces@faceListCoords[[ind]], 1];
    trinormals = First@
        Region`Mesh`MeshCellNormals[DiscretizeGraphics@*Graphics3D@#, 2] & /@ triangles;
    tricents = RegionCentroid@*DiscretizeGraphics@*Graphics3D /@ triangles;
    arrows = MapThread[Arrow[{#1, #1 + 0.5 #2}] &, {tricents, trinormals}];
    Graphics3D[{{Opacity[0.1], Blue, triangles}, Red, arrows}, ImageSize → Small]] /@
   Range[400];
```

In[ ]:=
```
ListAnimate[plts, SaveDefinitions → True]
```

Out[ ]=



```
(*DumpSave[
  "C:\\Users\\aliha\\Desktop\\wolfram-vertex-3D\\create geometry\\infinitesheet.mx",
  {edges,indToPtsAssoc,ptsToIndAssoc,xLim,yLim,cellVertexGrouping,vertexToCell}];*)
```