Polyhedral Models

Hassan Eshaq

Worcester Polytechnic Institute
Worcester, MA 01609-2280

Abstract

Consider a polyhedral surface in three-space that has the property that it can change its shape while keeping all its polygonal faces congruent. Adjacent faces are allowed to rotate along common edges. Mathematically exact flexible surfaces were found by Connelly in 1978. But the question remained as to whether the volume bounded by such surfaces was necessarily constant during the flex. In other words, is there a mathematically perfect bellows that actually will exhale and inhale as it flexes? For the known examples, the volume did remain constant. Following an idea of Sabitov, but using the theory of places in algebraic geometry (suggested

by Steve Chase), Connelly et al. showed that there is no perfect mathematical bellows. All flexible surfaces must flex with constant volume.

We built several models to illustrate the above theory. We start out by building stick models of the platonic solids, the tetrahedron, the octahedron, the cube, the dodecahedron and the icosahedron. In the regular case, i.e. when all the edges have the same length, the tetrahedron, the octahedron and the icosahedron are rigid, but the cube and the dodecahedron are not. The tetrahedron remains rigid when the edge lengths are altered, but the octahedron becomes flexible for certain choices of lengths. When the faces of these polyhedra are put in as solid surface, they all become rigid, as long as the faces can actually be fit in without the edges intersecting the faces. For the flexible octahedron, it is not possible to fit in faces. However, a model was built using a construction by Connelly. The model, with all nonpermeable material and tightly closed seams at the edges, flexes without bulging of the faces. Any attempt to move the platnonic solids leads to buckeling of the faces. Finally we built a model of the cubeoctahedron after a suggestion by Walser. This model is cut at a line of symmetry, pops up to minimize its energy stored by 4 rubber bands in its interior, and in doing so also maximizes its volume.

Three MATLAB files were created. The first one shows how the coboctahedron is obtained from either the cube or the octahedron by truncation.

The second one shows the motion of Walser's cubeoctahedron just as the physical model did move.

Thie third MATLAB file shows the theoretical motion of the object if selfintersection were possible.

1 Walser's pop-up Cuboctahedron

Program 1

```
function [] = cubeocty;
% This file creates the animations of the
% balence split cubeoctahedron.
%
% Note:  It is necessary to assign lights the
%        figure, and use something other than
%        flat shading.  Otherwise the MATLAB
%        program is confused and the order of the faces is
%        not correct.
%
% April 1, 2002.

% Clear any existing matlab figures.
delete(findobj('type','figure'));

% Set up the figure
figure('color','black');
%background color assigned black
```

```
cubeaxes = axes('tag','cubeaxes');
set(cubeaxes,'view',[32 23],...
       'cameraposition',[8.44883 -13.5209 6.76766],...
       'camerapositionmode','manual',...
       'CameraViewAngle',[9.62943],...
        'CameraViewAngleMode','manual')
light1 = light('Position',[-0.181186 -0.431186 0.883883],...
                'Color',[1 1 1],...
              'Style','infinite');
light2 = light('Position',[2 2 2],'Color',[1 1 1],...
                 'Style','infinite');
light3 = light('Position',[0 0 -5],'Color',[1 1 1],...
               'Style','infinite');
axis off
axis equal


% These are initial values for testing the figure.
h = 1;
% h is the distance from the top of the figure
% to the center of mass.
k = 1;
% k is one half the horizontal diameter of the figure.
p = 1;
%(p,q,0) is the coordinates of the first octant representative
% of the vertex of one isosceles right triangle.
q =   0;
h = .85
% V is the list of vertex coordinates.
V = [ [   h, 1-h,    0];...
      [   h,   0,  1-h];...
      [   h, h-1,    0];...
      [   h,   0,  h-1];...
      [  -h, 1-h,    0];...
      [  -h,   0,  1-h];...
      [  -h, h-1,    0];...
      [  -h,   0,  h-1];...
      [   0,   h,  1-h];...
      [ 1-h,   h,   0 ];...
      [   0,   h, h-1 ];...
      [ h-1,   h,   0 ];...
      [   0,  -h, 1-h ];...
      [ 1-h,  -h,   0 ];...
      [   0,  -h, h-1 ];...
      [ h-1,  -h,   0 ];...
      [ 1-h,    0,  h ];...
```

```
       [   0,  1-h,  h ];...
       [ h-1,    0,  h ];...
       [   0,  h-1,  h ];...
       [ 1-h,    0, -h ];...
       [   0,  1-h, -h ];...
       [ h-1,    0, -h ];...
       [   0,  h-1, -h ];...
       [   0,    0,  0 ]];

% T is the face matrix.
% Every row says which of the alligning
% vertices form a triangle.
S = [[1,1,1,2,3,4];...   % Top and Bottom squares (2)
     [5,5,5,6,7,8];...
     [9,9,9,10,11,12];...
     [13,13,13,14,15,16];...
     [17,17,17,18,19,20];...
     [21,21,21,22,23,24];...
     [2,1,10,9,18,17];...
     [2,3,14,13,20,17];...
     [4,1,10,11,22,21];...
     [3,4,21,24,15,14];...
     [9,12,5,6,19,18];...
     [11,12,5,8,23,22];...
     [8,7,16,15,24,23];...
     [7,6,19,20,13,16]];

% The figure is mostly gray, with a hint of color
% to assist the lighting.
SColor = rand(size(S,1),3);
%TColor([1:4],:) = TColor([1:4],:) - .5;
%TColor([13:28],:) = TColor([13:28],:) + .2*rand(16  ,3  );


cornerpatch = patch('vertices',V,'faces',S,...
     'FaceVertexCdata',SColor);
set(cornerpatch,'facecolor','flat');
set(cornerpatch,'facelighting','phong');


FrameNumber = 0;


for h = [[1:-.02:.5],[.5:.02:1]]
```

```
% This are the new vertex positions
V = [ [   h, 1-h,    0];...
      [   h,   0,  1-h];...
      [   h, h-1,    0];...
      [   h,   0,  h-1];...
      [  -h, 1-h,    0];...
      [  -h,   0,  1-h];...
      [  -h, h-1,    0];...
      [  -h,   0,  h-1];...
      [   0,   h,  1-h];...
      [ 1-h,   h,   0 ];...
      [   0,   h, h-1 ];...
      [ h-1,   h,   0 ];...
      [   0,  -h, 1-h ];...
      [ 1-h,  -h,   0 ];...
      [   0,  -h, h-1 ];...
      [ h-1,  -h,   0 ];...
      [ 1-h,    0,  h ];...
      [   0,  1-h,  h ];...
      [ h-1,    0,  h ];...
      [   0,  h-1,  h ];...
      [ 1-h,    0, -h ];...
      [   0,  1-h, -h ];...
       [ h-1,    0, -h ];...
      [   0,  h-1, -h ];...
      [ 0, 0,  0]]*1.5;

set(cornerpatch,'vertices',V);
% reassigns the vertex coordinates.
set(cubeaxes,'view',[32 23],...
    'cameraposition',[8.44883 -13.5209 6.76766],...
    'camerapositionmode','manual');
drawnow;

Mov(FrameNumber+1) = getframe(gcf);
% put the current figure in a movie (framelist).
FrameNumber = FrameNumber+1;
end;

% create the avi file.
movie2avi(Mov,'cubeocty.avi','compression','none');
```

Program 2

```
% This file creates the animations of
% the balence split cubeoctahedron.
%
% Note:  It is necessary to assign lights the the figure,
%        and use something other than
%        flat shading.  Otherwise the MATLAB program is
%        confused and the order of the faces is
%        not correct.
%
% April 1, 2002.

% Clear any existing matlab figures.
delete(findobj('type','figure'));

% Set up the figure
figure
cubeaxes = axes('tag','cubeaxes');
set(cubeaxes,'view',[32 23],'cameraposition',...
       [8.44883 -13.5209 6.76766],...
       'camerapositionmode','manual',...
```

```
       'CameraViewAngle',[9.62943],...
       'CameraViewAngleMode','manual')
light1 = light('Position',[-0.181186 -0.431186 0.883883],...
       'Color',[1 1 1],'Style','infinite');
light1 = light('Position',[2 2 2],'Color',...
      [1 1 1],'Style','infinite');
axis off
axis equal


% These are initial values for testing the figure.
h = 1;
% h is the height of the top of the figure
%from the center of mass.
k = 1;
% k is one half the horizontal diameter of the figure.
p = 1;
%(p,q,0) is the coordinates of the first
% octant representative
% of the vertex of one isosceles right triangle.
q = 0;

% V is the list of vertex coordinates.
V = [ [ 1,  0,  h];...
      [ 0,  1,  h];...
      [-1,  0,  h];...
      [ 0, -1,  h];...
      [ 1,  0, -h];...
      [ 0,  1, -h];...
      [-1,  0, -h];...
      [ 0, -1, -h];...
      [ k,  k,  0];...
      [-k,  k,  0];...
      [-k, -k,  0];...
      [ k, -k,  0];...
      [ p,  q,  0];...
      [ q,  p,  0];...
      [-p,  q,  0];...
      [-q,  p,  0];...
      [-p, -q,  0];...
      [-q, -p,  0];...
      [ p, -q,  0];...
      [ q, -p,  0]]

% T is the face matrix.  Every row says which of the
% alligning vertices form a triangle.
```

```
T = [[1,2,3];...   % Top and Bottom squares (2)
     [1,4,3];...
     [5,6,7];...
     [5,8,7];...
     [1 2 9];...   % Equilateral Triangles. (8)
     [2 3 10];...
     [3 4 11];...
     [4 1 12];...
     [5 6 9];...
     [6 7 10];...
     [7 8 11];...
     [8 5 12];...
     [1,9,13];...  % Isosceles Right Triangles (16)
     [5,9,13];...
     [2,9,14];...
     [6,9,14];...
     [3,10,15];...
     [7,10,15];...
     [2,10,16];...
     [6,10,16];...
     [3,11,17];...
     [7,11,17];...
     [4,11,18];...
     [8,11,18];...
     [1,12,19];...
     [5,12,19];...
     [4,12,20];...
     [8,12,20]];

% The figure is mostly gray, with a hint of color to
% assist the lighting.
TColor = .5+0*T;
size(TColor)
TColor([1:4],:) = TColor([1:4],:) - .5;
TColor([13:28],:) = TColor([13:28],:) + .2*rand(16  ,3  );


cornerpatch = patch('vertices',V,'faces',T,...
     'FaceVertexCdata',TColor);
set(cornerpatch,'facecolor','flat');
set(cornerpatch,'facelighting','phong');


FrameNumber = 0;
% The figure parameters, k, p and q, will all depend on h.
```

```
for h = [[1:-.02:0],[0:.02:1]]

% The distance between [1 0 h] and [k k 0] must be sqrt(2);
% So (1-k^2) + k^3 + h^2 = 2;
%h = sqrt(2-k^2 - (k-1)^2);
k = (1/2)+sqrt((1/2)*((3/2)-h^2));
% In the first quadrant the vertices of the vertices
% are [1 0 h] and [k k 0].
% They are the hypotenuse of an isosceles right triangle.
% The third point is [p q 0],
%since it lies on the x-y plane.
% It must have distance 1 from both [1 0 h] and [k k 0].
% So (p-1)^2 + q^2 + h^2 = 1 and
%    (p-k)^2 + (q-k)^2 = 1.
% So
%  p^2  -2p   + 1    + q^2 + h^2 = 1
%  p^2  - 2pk + k^2  + q^2 - 2qk + k^2 = 1
% After subtracting which can express q = a + bp
% so we can find first p, then q.

a = (2*k^2 - h^2 - 1)/(2*k);
b = (1-k)/k;
% Now Ap^2 + Bp + C = 0;
A = 1 + b^2;
B = 2*(a*b - 1);
C = h^2 + a^2;
% the collapsing mode is expressed in the
% sign of the square root.
p = (-B + sqrt(B^2 - 4*A*C))/(2*A);
q = sqrt(1 - h^2 - (1-p)^2);

% This are the new vertex positions
V = [ [ 1,  0,  h];...
      [ 0,  1,  h];...
      [-1,  0,  h];...
      [ 0, -1,  h];...
      [ 1,  0, -h];...
      [ 0,  1, -h];...
      [-1,  0, -h];...
      [ 0, -1, -h];...
      [ k,  k,  0];...
      [-k,  k,  0];...
      [-k, -k,  0];...
      [ k, -k,  0];...
```

```
      [ p,  q,  0];...
      [ q,  p,  0];...
      [-p,  q,  0];...
      [-q,  p,  0];...
      [-p, -q,  0];...
      [-q, -p,  0];...
      [ p, -q,  0];...
      [ q, -p,  0]];

set(cornerpatch,'vertices',V);
% reassigns the vertex coordinates.
set(cubeaxes,'view',[32 23],'cameraposition',...
      [8.44883 -13.5209 6.76766],...
      'camerapositionmode','manual');
drawnow;

Mov(FrameNumber+1) = getframe(gcf);
% put the current figure in a movie (framelist).
FrameNumber = FrameNumber+1;
end;

% create the avi file.
movie2avi(Mov,'cubey.avi','compression','none');
```

Program 3

```
% This file creates the animations of the
% split cubeoctahedron.
%
% This version is modified to show both modes
%
% Note:  It is necessary to assign lights the the figure,
%        and use something other than
%        flat shading.  Otherwise the MATLAB program is
%        confused and the order of the faces is
%        not correct.
%
% April 1, 2002.

% Clear any existing matlab figures.
delete(findobj('type','figure'));

% Set up the figure
figure
cubeaxes = axes('tag','cubeaxes');
set(cubeaxes,'view',[32 23],'cameraposition',...
```

```
       [8.44883 -13.5209 6.76766],...
       'camerapositionmode','manual',...
       'CameraViewAngle',[9.62943],...
       'CameraViewAngleMode','manual')
light1 = light('Position',[-0.181186 -0.431186 0.883883],...
               'Color',[1 1 1],'Style','infinite');
light1 = light('Position',[2 2 2],'Color',...
          [1 1 1],'Style','infinite');
axis off
axis equal


% These are initial values for testing the figure.
h = 1;
% h is the height of the top of the figure
% from the center of mass.
k = 1;
% k is one half the horizontal diameter of the figure.
p = 1;
%(p,q,0) is the coordinates of the first octant representative
% of the vertex of one isosceles right triangle.
q = 0;

% V is the list of vertex coordinates.
V = [ [ 1,  0,  h];...
      [ 0,  1,  h];...
      [-1,  0,  h];...
      [ 0, -1,  h];...
      [ 1,  0, -h];...
      [ 0,  1, -h];...
      [-1,  0, -h];...
      [ 0, -1, -h];...
      [ k,  k,  0];...
      [-k,  k,  0];...
      [-k, -k,  0];...
      [ k, -k,  0];...
      [ p,  q,  0];...
      [ q,  p,  0];...
      [-p,  q,  0];...
      [-q,  p,  0];...
      [-p, -q,  0];...
      [-q, -p,  0];...
      [ p, -q,  0];...
      [ q, -p,  0]]

% T is the face matrix.
```

```
% Every row says which of the alligning vertices form a triangle.
T = [[1,2,3];...   % Top and Bottom squares (2)
     [1,4,3];...
     [5,6,7];...
     [5,8,7];...
     [1 2 9];...   % Equilateral Triangles. (8)
     [2 3 10];...
     [3 4 11];...
     [4 1 12];...
     [5 6 9];...
     [6 7 10];...
     [7 8 11];...
     [8 5 12];...
     [1,9,13];...  % Isosceles Right Triangles (16)
     [5,9,13];...
     [2,9,14];...
     [6,9,14];...
     [3,10,15];...
     [7,10,15];...
     [2,10,16];...
     [6,10,16];...
     [3,11,17];...
     [7,11,17];...
     [4,11,18];...
     [8,11,18];...
     [1,12,19];...
     [5,12,19];...
     [4,12,20];...
     [8,12,20]];

% The figure is mostly gray, with a hint of color to
% assist the lighting.
TColor = .5+0*T;
size(TColor)
TColor([1:4],:) = TColor([1:4],:) - .5;
TColor([13:28],:) = TColor([13:28],:) + .2*rand(16  ,3  );


cornerpatch = patch('vertices',V,'faces',T,...
  'FaceVertexCdata',TColor);
set(cornerpatch,'facecolor','flat');
set(cornerpatch,'facelighting','phong');


FrameNumber = 0;
% The figure parameters, k, p and q, will all depend on h.
```

```
for h = [0:.04:1]

% The distance between [1 0 h] and [k k 0] must be sqrt(2);
% So (1-k^2) + k^3 + h^2 = 2;
%h = sqrt(2-k^2 - (k-1)^2);
k = (1/2)+sqrt((1/2)*((3/2)-h^2));
% In the first quadrant the vertices of the vertices
% are [1 0 h] and [k k 0].
% They are the hypotenuse of an isosceles right triangle.
% The third point is [p q 0], since it lies on the x-y plane.
% It must have distance 1 from both [1 0 h] and [k k 0].
% So (p-1)^2 + q^2 + h^2 = 1 and
%    (p-k)^2 + (q-k)^2 = 1.
% So
%  p^2  -2p   + 1    + q^2 + h^2 = 1
%  p^2  - 2pk + k^2  + q^2 - 2qk + k^2 = 1
% After subtracting which can express q = a + bp
% so we can find first p, then q.

a = (2*k^2 - h^2 - 1)/(2*k);
b = (1-k)/k;
% Now Ap^2 + Bp + C = 0;
A = 1 + b^2;
B = 2*(a*b - 1);
C = h^2 + a^2;
% the collapsing mode is expressed in the
%sign of the square root.
p = (-B + sqrt(B^2 - 4*A*C))/(2*A);
q = sqrt(1 - h^2 - (1-p)^2);

% This are the new vertex positions
V = [ [ 1,  0,  h];...
      [ 0,  1,  h];...
      [-1,  0,  h];...
      [ 0, -1,  h];...
      [ 1,  0, -h];...
      [ 0,  1, -h];...
      [-1,  0, -h];...
      [ 0, -1, -h];...
      [ k,  k,  0];...
      [-k,  k,  0];...
      [-k, -k,  0];...
      [ k, -k,  0];...
```

```
      [ p,  q,  0];...
      [ q,  p,  0];...
      [-p,  q,  0];...
      [-q,  p,  0];...
      [-p, -q,  0];...
      [-q, -p,  0];...
      [ p, -q,  0];...
      [ q, -p,  0]];

set(cornerpatch,'vertices',V);
% reassigns the vertex coordinates.
set(cubeaxes,'view',[32 23],'cameraposition',...
     [8.44883 -13.5209 6.76766],'camerapositionmode','manual');
drawnow;

Mov(FrameNumber+1) = getframe(gcf);
% put the current figure in a movie (framelist).
FrameNumber = FrameNumber+1;
end;

for h = [[1:-.04:0],[0:.04:1]]

% The distance between [1 0 h] and [k k 0] must be sqrt(2);
% So (1-k^2) + k^3 + h^2 = 2;
%h = sqrt(2-k^2 - (k-1)^2);
k = (1/2)+sqrt((1/2)*((3/2)-h^2));
% In the first quadrant the vertices of the vertices
% are [1 0 h] and [k k 0].
% They are the hypotenuse of an isosceles right triangle.
% The third point is [p q 0], since it lies on the x-y plane.
% It must have distance 1 from both [1 0 h] and [k k 0].
% So (p-1)^2 + q^2 + h^2 = 1 and
%    (p-k)^2 + (q-k)^2 = 1.
% So
%  p^2  -2p   + 1    + q^2 + h^2 = 1
%  p^2  - 2pk + k^2  + q^2 - 2qk + k^2 = 1
% After subtracting which can express q = a + bp
% so we can find first p, then q.

a = (2*k^2 - h^2 - 1)/(2*k);
b = (1-k)/k;
% Now Ap^2 + Bp + C = 0;
A = 1 + b^2;
B = 2*(a*b - 1);
C = h^2 + a^2;
% the collapsing mode is expressed in the
```

```
%sign of the square root.
p = (-B - sqrt(B^2 - 4*A*C))/(2*A);
q = sqrt(1 - h^2 - (1-p)^2);

% This are the new vertex positions
V = [ [ 1,  0,  h];...
      [ 0,  1,  h];...
      [-1,  0,  h];...
      [ 0, -1,  h];...
      [ 1,  0, -h];...
      [ 0,  1, -h];...
      [-1,  0, -h];...
      [ 0, -1, -h];...
      [ k,  k,  0];...
      [-k,  k,  0];...
      [-k, -k,  0];...
      [ k, -k,  0];...
      [ p,  q,  0];...
      [ q,  p,  0];...
      [-p,  q,  0];...
      [-q,  p,  0];...
      [-p, -q,  0];...
      [-q, -p,  0];...
      [ p, -q,  0];...
      [ q, -p,  0]];

set(cornerpatch,'vertices',V);
% reassigns the vertex coordinates.
set(cubeaxes,'view',[32 23],'cameraposition',
...[8.44883 -13.5209 6.76766],'camerapositionmode','manual');
drawnow;

Mov(FrameNumber+1) = getframe(gcf);
% put the current figure in a movie (framelist).
FrameNumber = FrameNumber+1;
end;

for h = [1:-.04:0]

% The distance between [1 0 h] and [k k 0] must be sqrt(2);
% So (1-k^2) + k^3 + h^2 = 2;
%h = sqrt(2-k^2 - (k-1)^2);
k = (1/2)+sqrt((1/2)*((3/2)-h^2));
% In the first quadrant the vertices of the vertices
% are [1 0 h] and [k k 0].
% They are the hypotenuse of an isosceles right triangle.
```

```
% The third point is [p q 0], since it lies on the x-y plane.
% It must have distance 1 from both [1 0 h] and [k k 0].
% So (p-1)^2 + q^2 + h^2 = 1 and
%    (p-k)^2 + (q-k)^2 = 1.
% So
%  p^2  -2p   + 1    + q^2 + h^2 = 1
%  p^2  - 2pk + k^2  + q^2 - 2qk + k^2 = 1
% After subtracting which can express q = a + bp
% so we can find first p, then q.

a = (2*k^2 - h^2 - 1)/(2*k);
b = (1-k)/k;
% Now Ap^2 + Bp + C = 0;
A = 1 + b^2;
B = 2*(a*b - 1);
C = h^2 + a^2;
% the collapsing mode is expressed in the
% sign of the square root.
p = (-B + sqrt(B^2 - 4*A*C))/(2*A);
q = sqrt(1 - h^2 - (1-p)^2);

% This are the new vertex positions
V = [ [ 1,  0,  h];...
      [ 0,  1,  h];...
      [-1,  0,  h];...
      [ 0, -1,  h];...
      [ 1,  0, -h];...
      [ 0,  1, -h];...
      [-1,  0, -h];...
      [ 0, -1, -h];...
      [ k,  k,  0];...
      [-k,  k,  0];...
      [-k, -k,  0];...
      [ k, -k,  0];...
      [ p,  q,  0];...
      [ q,  p,  0];...
      [-p,  q,  0];...
      [-q,  p,  0];...
      [-p, -q,  0];...
      [-q, -p,  0];...
      [ p, -q,  0];...
      [ q, -p,  0]];

set(cornerpatch,'vertices',V);
% reassigns the vertex coordinates.
set(cubeaxes,'view',[32 23],'cameraposition',...
```

```
      [8.44883 -13.5209 6.76766],...
      'camerapositionmode','manual');
drawnow;
Mov(FrameNumber+1) = getframe(gcf);
% put the current figure in a movie (framelist).
FrameNumber = FrameNumber+1;
end;
% create the avi file.
movie2avi(Mov,'cubey.avi','compression','none');
```

2 The cuboctahedron as limit of a permutation graph

The cubeoctahedron described in the previous section can also be thought of as a limiting case of a truncated octahedron. We first cut off a bit of the corners of the octahedron perpendicular to the diagonals. Every vertex of the original octahedron is replaced by a square, so the resulting polyhedron has 24 vertices and 36 edges, 6 faces which are squares and 8 face s which are hexagons. We can cut off more and more of the corners until the hexagons become regular. The underlying graph of this polyhedron us usually called a permutation graph for the permutations of 4 elements, where two permutations are adjacent if they differ by a transposition of adjacent elements. Clearly, if we have a permutation of 4 elements, (a_1, a_2, a_3, a_4), there are 3 adjacent transpositions, namely switching a_1 and a_2, switching a_2 and a_3 or switching a_3 and a_4. We denote these in cycle notation by (12), (23), and (34). Since (12) and (23) commute, the permutation graph has quatrilaterals. (23) does not commute with (12) or (34), but the product of (12)(23)(12)(23)(12)(23) is the identity, similarly with (12) replaced by (34). Therefore the permutation graph also contains hexagons. It is clear that (12) and (34) play equivalent roles, but (23) is different. Contracting the edges corresponding to (23) in the permutation graph identifies pairs of vertices (permutations) that differ by a transposition of the middle digits. When the contraction is complete, the resulting graph has 24/2=12 vertices. The hexagons are now triangles, but all the original quatrilaterals are still in tact.

Now we ask for an interpretation of the resulting polyhedron. First we observe that the polyhedron obtained from shrinking the edges corresponding to (23) is an Archemedian solid, namely the cubeoctahedron, which we considered in the previous chapter.Now we want to see what interpretation we can get from the labelling. If we omit the middle two numbers from the permutations which were the labelling of the permutation graph, we get permutations of length two, each of which is listed twice. Identifying equal permutations, we get a labelling of the vertices of the cubeoctahedron.

Starting with the permutations of length 2 on 4 elements, we can define two of these permutations to be close or adjacent if they only differ in one entry, but

match in the other. That way, every vertex (a,b) is adjacent to 4 other vertices because, leaving b fixed, there are two choices different from a, namely the 4 elemets except a or b, and if we fix a, we get two choices as well. Triangles in this graph are keeping one entry fixed and the second entry is changed over all three possibilities. Alltogether we obtain 8 triangles, namely (1,2), 1,3), 1,4); (2,1), (2,3), (2,4); (3,1), (3,2),(3,4); (4,1),(4,2), (4,3); (2,1), (3,1), (4,1); (1,2), (3,2), (4,2); (1,3),(2,3), (4,3); and (1,4), (2,4), (3,4). We also get the following six quadrilaterals: (1,2), (1,3), (4,3), (4,2); (1,2), (1,4), (3,4), (3,2); (1,3), (2,3), (2,4), (1,4); (2,4), (2,1), (3,1), (3,4); (2,3), (2,1), (4,1), (4,3); (3,1), (3,2), (4,2), (4,1).

References

[1] Robert Connelly, *The rigidity of polyhedral Surfaces,* Mathematics Magazine Vol. 52, No. 5, 275–283, 1979

[2] Hans Walser, *The pop-up Cuboctahedron,* The College Mathematics Journal Vol. 3, No. 2, March 2000

[3] Jack Graver, Brigitte Servatius, Herman Servatius, *Combinatorial Rigidity,* Graduate Studies in Mathematics, Vol. 2, AMS, 1993

[4] Robert Connelly, I. Sabitov and A. Walz, *The Bellows Conjecture,* Contributions to Algebra and Geometry , Vol. 38 (1997), No.1, 1-10.