**NATIONAL TECHNICAL UNIVERSITY OF UKRAINE**

**“KYIV POLYTECHNIC INSTITUTE”**

**FACULTY OF HEAT POWER ENGINEERING**

**DEPARTMENT OF AUTOMATION OF POWER PROCESSES AND**

**SYSTEMS ENGINEERING**

# COURSEWORK

course: “Visualization of graphical and geometrical information”

theme: “Tangents and a normal to a point lying on the shoe surface”

Variant 12

Presented to:

Anatolii Demchyshyn

Prepared by:

Dmytro Kruhlyi

5th year student

Group TR-02mp

Kyiv – 2020

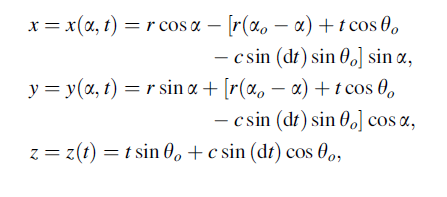
**Task:**

* Calculate tangents in a point on a surface using either numerical or numeric partial derivatives.
* Calculate a normal vector taking cross product of the two derivatives.
* It has to be possible to move the point along the surface *(u,v)* space using a keyboard. E.g. keys **A** and **D** move the point along *u* parameter and keys **W** and **S** move the point along *v* parameter.

**Theory:**

A Monge surface with a cylindrical directrix surface and a sinusoid curve as meridian may be defined as a surface formed by a moving sinusoid, the plane of which rolls without slipping above a right circular cylinder. Assuming a sinusoid as a generatrix curve or a meridian and an evolvent of the circle as a directrix curve or a parallel, it is possible to give another determination for a studied Monge surface: Monge surface with a cylinder directrix surface and a sinusoidal generatrix is formed by a plane sinusoid moving along a directrix evolvent of the circle, so that a plane with the generatrix sinusoid lies all the time at the normal plane of directrix evolvent of the circle and is deadly linked with it.

*Parametrical equations (Fig. 1):*



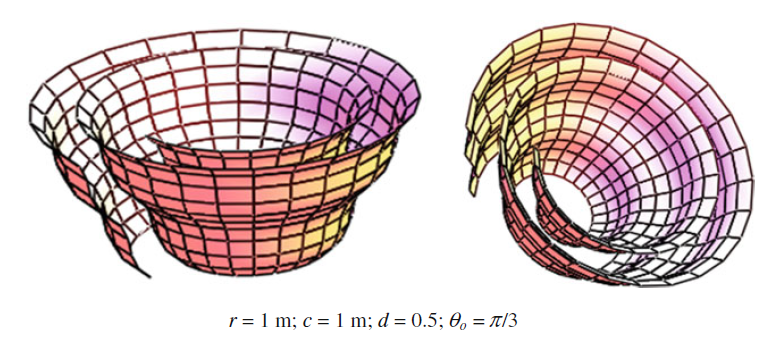


Fig. 1

The surface is given in lines of the principal curvatures t, α. One family of the plane lines of principal curvatures denoted by t coincides with the generatrix sinusoids, but the second family of the plane lines of principal curvatures α is the family of the evolvents of the circle with a radius r.

To get tangents vectors we can use numeric partial derivative:

To get a normal vector we can use inner product :

**Implementation:**

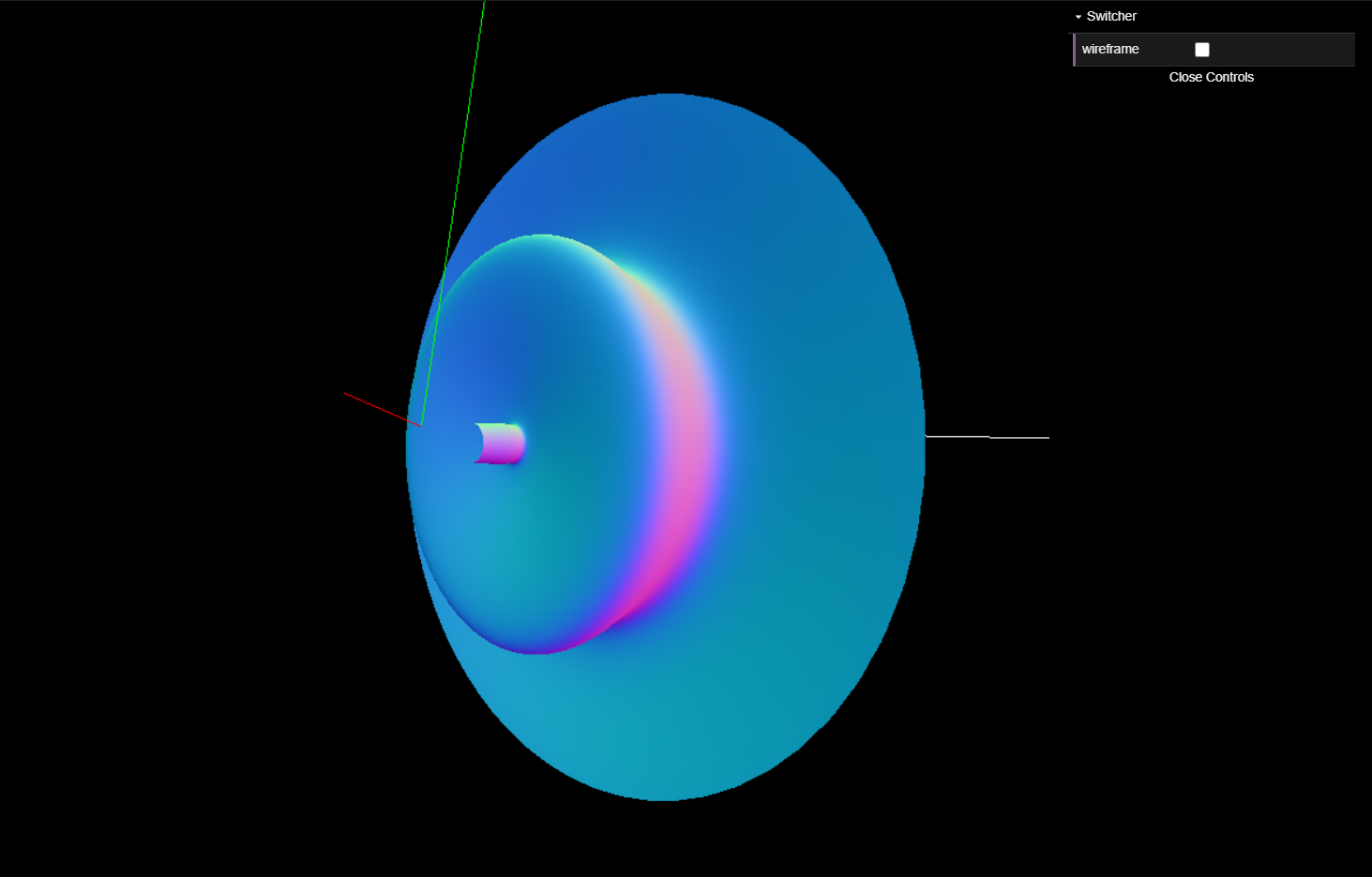
For implementation this surface and tangents with normal I used JavaScript language and three.js library. Three.js library allow us to get 3D content on a webpage and it’s very easy to use it.

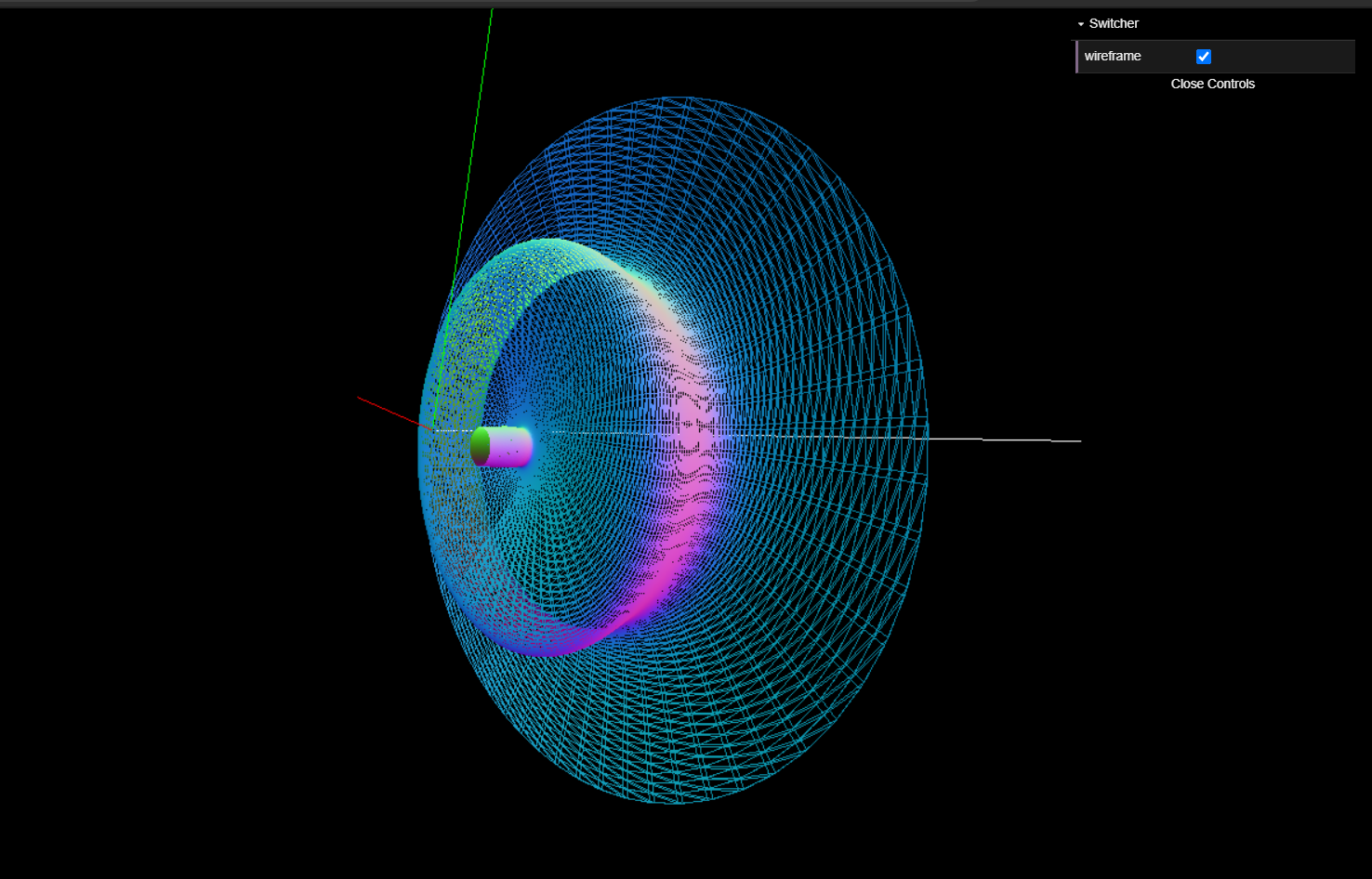
The process of rendering consists of 7 parts:

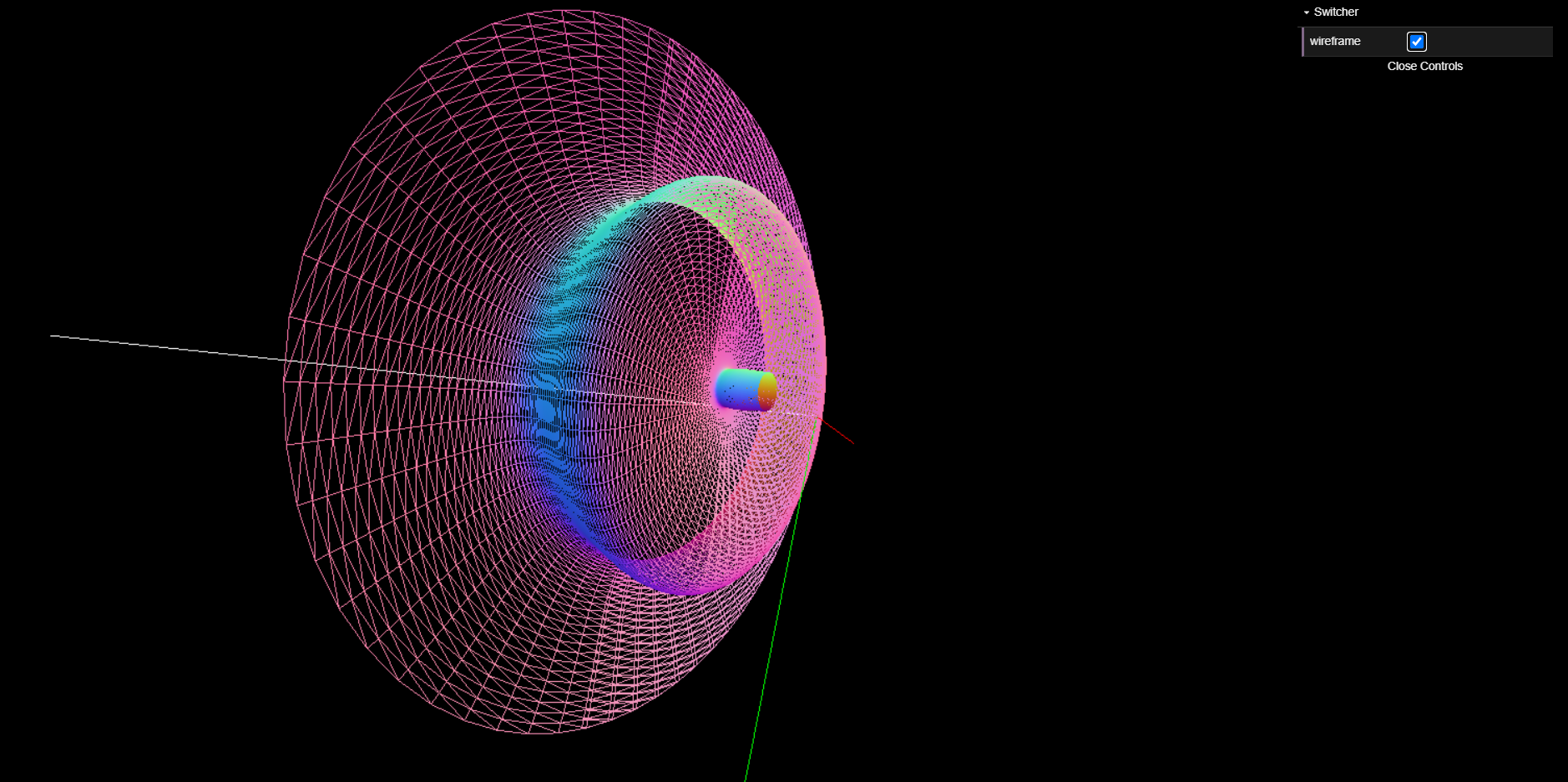
1. Create a scene and a camera using THREE.PerspectiveCamera and THREE.Scene classes.
2. Create custom *mongeSurfaceDiretrixSinusoidaMeridian* function and new geometry using THREE.ParametricGeometry. Add it to new THREE.Mesh and add this mesh to scene.
3. Create functions which use numeric partial derivatives to calculate tangents vectors.
4. Create a function to calculate a normal vector.
5. Create three objects of THREE.LineBasicMaterial to visualize tangents vectors and a normal.
6. Add Event Listener to the document to move dots on a surface using keys “A”, “W”, “S” and “D”. Кeys “A” and “D” move the point along *u* parameter and keys “W” and “S” move the point along *v* parameter. Point movement is limited to the limits of the function.
7. Render scene using THREE.WebGLRenderer.

**Screenshots:**

The screenshots show an example of point movement. The tangent U vector is green, the tangent V vector is red and the normal vector is white.







**Source code:**

The source code of my program is shown below.

const canvas = document.getElementById("canvas");

const scene = new THREE.Scene();

const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);

camera.position.set(0, 0, -10);

camera.lookAt(0, 0, 0);

const renderer = new THREE.WebGLRenderer({ canvas: canvas });

renderer.setSize(window.innerWidth, window.innerHeight);

renderer.setClearColor(0x000000);

const delta = 0.001;

let mainNormal, firstTangent, secondTangent;

const geometry = new THREE.ParametricGeometry(mongeSurfaceDiretrixSinusoidaMeridian, 100, 100);

const material = new THREE.MeshNormalMaterial();

const kleinBottle = new THREE.Mesh(geometry, material);

scene.add(kleinBottle);

const gui = new dat.GUI();

const box = gui.addFolder('Switcher');

box.add(kleinBottle.material, 'wireframe').listen();

box.open();

var movingU = 0.5;

var movingV = 0.5;

const controls = new THREE.OrbitControls(camera, renderer.domElement);

const light = new THREE.AmbientLight(0xffffff);

scene.add(light);

drawingTangentsAndNormal(movingU, movingV);

renderer.setAnimationLoop(() => { renderer.render(scene, camera); })

document.addEventListener('keydown',(event) => {

  event.preventDefault();

      switch (event.code) {

        case "KeyA":

          movingU -= 0.01;

          drawingTangentsAndNormal(movingU, movingV);

          break;

        case "KeyD":

          movingU += 0.01;

          drawingTangentsAndNormal(movingU, movingV);

          break;

        case "KeyW":

          movingV += 0.01;

          drawingTangentsAndNormal(movingU, movingV);

          break;

        case "KeyS":

          movingV -= 0.01;

          drawingTangentsAndNormal(movingU, movingV);

          break;

    }

});

function drawingTangentsAndNormal(alpha, t) {

  alpha \*= 10;

  t \*= 10;

  let tv = new THREE.Vector3(dxdv(alpha,t), dydv(alpha,t), dzdv(alpha,t));

  let tu = new THREE.Vector3(dxdu(alpha,t), dydu(alpha,t), dzdu(alpha,t));

  let normal = new THREE.Vector3(0,0,0).copy(tv).crossVectors(tv, tu);

  let vect = new THREE.Vector3(calcX(alpha, t), calcY(alpha, t), calcZ(alpha, t));

  let geomTv = new THREE.Geometry();

  geomTv.vertices.push(tv, vect);

  let geomTu = new THREE.Geometry();

  geomTu.vertices.push(tu, vect);

  let geomNormal = new THREE.Geometry();

  geomNormal.vertices.push(normal, vect);

  if(mainNormal == null) {

    firstTangent = new THREE.Line(geomTv, new THREE.LineBasicMaterial({ color: 0xFF0000 }));

    secondTangent = new THREE.Line(geomTu, new THREE.LineBasicMaterial({ color: 0x00FF00 }));

    mainNormal = new THREE.Line(geomNormal, new THREE.LineBasicMaterial({ color: 0xFFFFFF }));

  }

  firstTangent.geometry = geomTv;

  secondTangent.geometry = geomTu;

  mainNormal.geometry = geomNormal;

  scene.add(firstTangent);

  scene.add(secondTangent);

  scene.add(mainNormal);

}

function dxdv(u, v){

  return (calcX(u, v + delta \* v) - calcX(u, v)) / delta;

}

function dydv(u, v){

  return (calcY(u, v + delta \* v) - calcY(u, v)) / delta;

}

function dzdv(u, v){

  return (calcZ(u, v + delta \* v) - calcZ(u, v)) / delta;

}

function dxdu(u, v){

  return (calcX(u + delta \* u, v) - calcX(u, v)) / delta;

}

function dydu(u, v){

  return (calcY(u + delta \* u, v) - calcY(u, v)) / delta;

}

function dzdu(u, v){

  return (calcZ(u + delta \* u, v) - calcZ(u, v)) / delta;

}

function calcX(alpha, t){

  return  1 \* Math.cos(alpha) -

  (t \* Math.cos(Math.PI \* 0.2) - 2 \* Math.sin(1 \* t) \* Math.sin(Math.PI \* 0.2) + t \* Math.cos(Math.PI \* 0.2) -

  2 \*  Math.sin(1 \* t) \* Math.sin(Math.PI \* 0.2)) \* Math.sin(alpha);

}

function calcY(alpha, t){

  return 1 \* Math.sin(alpha) +

  (t \* Math.cos(Math.PI \* 0.2) - 2 \* Math.sin(1 \* t) \* Math.sin(Math.PI \* 0.2) + t \* Math.cos(Math.PI \* 0.2) -

  2 \*  Math.sin(1 \* t) \* Math.sin(Math.PI \* 0.2)) \* Math.cos(alpha);

}

function calcZ(alpha, t){

  return t \* Math.sin(Math.PI \* 0.2) + 2 \* Math.sin(1 \* t) \* Math.cos(Math.PI \* 0.2);

}

function mongeSurfaceDiretrixSinusoidaMeridian(alpha, t, target){

  alpha \*= 10;

  t \*= 10;

  let x = calcX(alpha, t);

  let y = calcY(alpha, t);

  let z = calcZ(alpha, t);

  return target.set(x, y, z);

}

**Conclusions:**

After implementation this course work I improved my skills with Three.js library and also learned to build tangents and normal to a given surface.