Interactive in 3D - Vector fields and Geometry

4. JSXGraph Conference 10.-12. October 2023

Wigand Rathmann

2023-10-12

Contents

1 Objective

Parametrized surface given as

$$S = \{\mathbf{s}(u,v) \,|\, (u,v) \in G \subset \mathbb{R}^2\} \subset \mathbb{R}^3.$$

The oriented curve integral can computed by

$$\int_S \langle \mathbf{V}(\mathbf{x}),\ \mathrm{d}S \rangle = \int_G \langle \mathbf{V}(\mathbf{s}(u,v))\,,\ (\mathbf{s}_{,u}(u,v) \times \mathbf{s}_{,v}(u,v))\,\rangle\ \mathrm{d}(u,v)$$

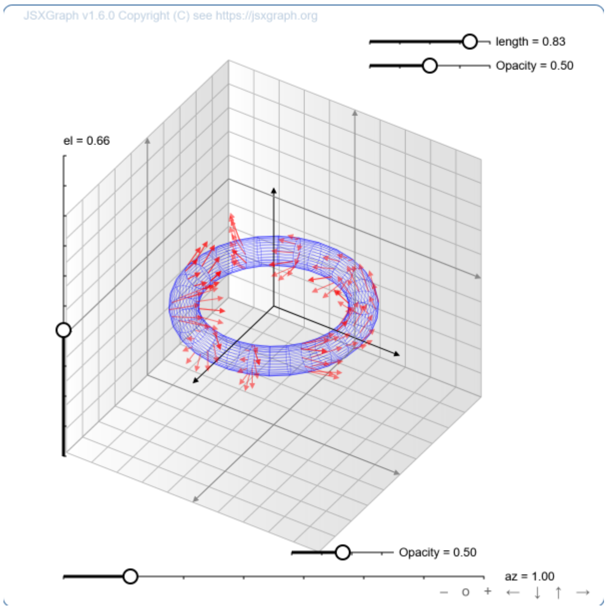

Figure 1: Surface and vector field

2 3D boards

2.1 Axis labels

3D view initialized with a box like

```
var box = [-2,2];
var view = board.create('view3d',
    [
        [-6, -3],
        [8, 8],
        [box, box, box]
    ])
```

The labels added depending on box.

```
var xlabel = view.create(
    "point3d",
    [0.9 * box[1], 0, 0.6 * box[0] + 0.4 * box[1]], {
        size: 0,
        name: "x"
    }
);
var Ylabel = view.create(
    "point3d",
    [0, 0.9 * box[1], 0.6 * box[0] + 0.4 * box[1]], {
```

```
        size: 0,
        name: "y"
    }
);
var Zlabel = view.create(
    "point3d",
    [
        0.7 * (0.6 * box[0] + 0.4 * box[1]),
        0.7 * (0.6 * box[0] + 0.4 * box[1]),
        0.9 * box[1],
    ], {
        size: 0,
        name: "z"
    }
);
```

3 Triple Product

The above integrand is $\langle \mathbf{V}(\mathbf{s}(u,v)) \,,\, (\mathbf{s}_{,u}(u,v) \times \mathbf{s}_{,v}(u,v)) \,\rangle$. Geometrically this is a triple product and we get the flow through the surface by the scalar product of the vector field an the normal vector on the surface times the area $\mathrm{d}S$. So the idea is explained geometrically.

The vectors $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^3$ define parallelepiped. The volume is given by $|\langle \mathbf{u}, \mathbf{v} \times \mathbf{w} \rangle|$. The idea can be explained in a

JSfiddle: https://jsfiddle.net/WigandR/bj3wx41e/

The applet shows not the whole construction of the triple product at once. Starting from a plane given by three points A, B and C in $\mathbb{R}^3$ one can unhide the normal of the plane, a forth point D, the visualisation of the parallepiped and the projection of D onto the normal.

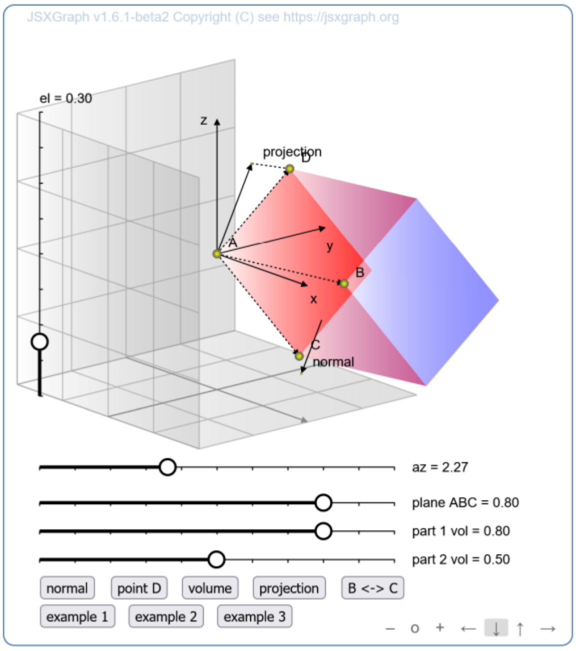

Figure 2: Triple Product

Dragging the points in the xy-plane by picking. In z-direction by pressing `shift` key.

3.1 Plane 3d

The points `A`, `B` and `C` are `point3d` objects. From then the difference (`direction`) and the length (`norm2`) are computed to feed the `plane3d` object creation call.

```
// ABC plane
var dirAB = direction(A, B);
var dirAC = direction(A, C);
var dirAD = direction(A, D);

var planeABC = view.create('plane3d', [
    A, dirAB, dirAC,
    [0, () => (norm2(dirAB))],
    [0, () => (norm2(dirAC))]
], {
    fillOpacity: () => opac1.Value(),
    fillColor: 'red'
});
```

The normal of a plane is accessible by the member `normal`. The functions direction and norm2 were implemented by the author inside the applet.

4 Vector fields

A very nice feature is the drawing of vector fields. In version 1.6.0 new objects `slopefield` and `vectorfield` have been introduced.

4.1 ODEs

4.1.1 1D

$f : \mathbb{R}^2 \to \mathbb{R}$ can be interpreted as the right hand side of an ODE $y'(x) = f(x, y)$. At each point in the xy-plane the slope is given and be plotted by `slopefield`

https://jsfiddle.net/WigandR/p6o4e3x9/

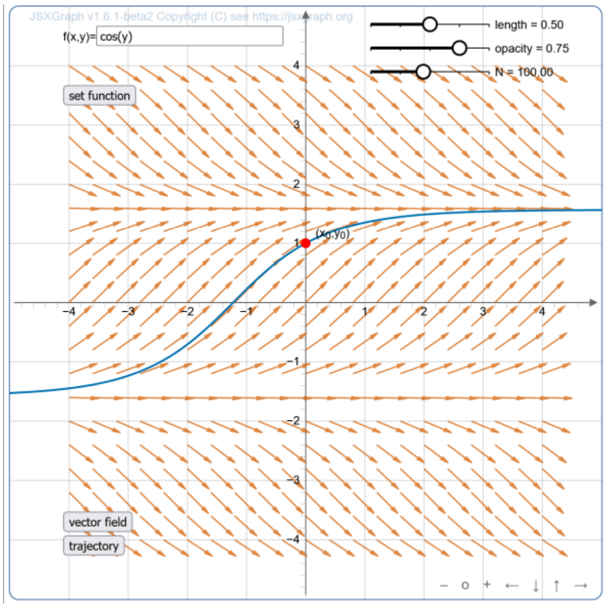

Figure 3: Slope field

In this applet the field is drawn by the command

```
field = board.create('slopefield', [f,
    [-4, 20, 4],
    [-4, 20, 4], 0.5
 ], {
    strokeWidth: 1.5,
    highlightStrokeWidth: 0.5,
    strokeColor: JXG.palette.red,
    strokeOpacity: () => o.Value(),
```

```
    highlightStrokeColor: JXG.palette.blue,
    scale: () => s.Value(),
    arrowHead: {
      enabled: true,
      size: 8,
      angle: Math.PI / 16
    },
    visible: true
  });
```

4.1.2 2D

Very close to this example is the case of $\mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^2$.
https://jsfiddle.net/WigandR/nwocugx2/

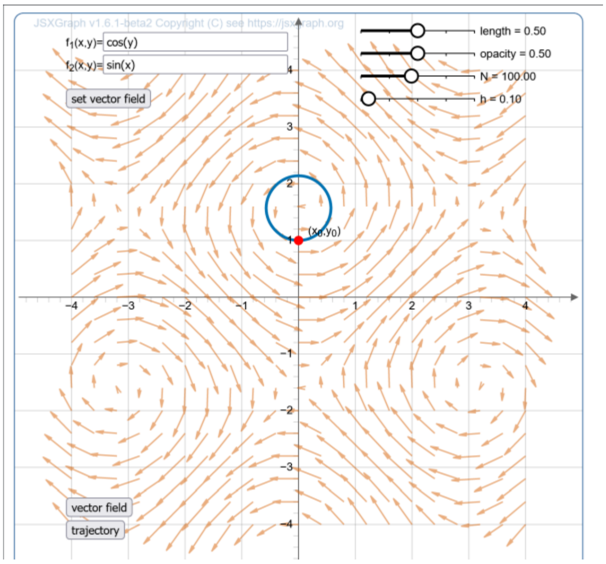

Figure 4: Vector field

At each point in the xy-plane the direction is given and be plotted by `vectorfield`

To integrate the vectorfield

```
fx = board.jc.snippet(inputfx.Value(), true, 'x,y');
fy = board.jc.snippet(inputfy.Value(), true, 'x,y');

field = board.create('vectorfield', [
  [fx, fy],
  [-4, 20, 4],
  [-4, 20, 4]
], {
```

```
    strokeWidth: 1.5,
    highlightStrokeWidth: 0.5,
    strokeColor: JXG.palette.red,
    strokeOpacity: () => o.Value(),
    highlightStrokeColor: JXG.palette.blue,
    scale: () => s.Value(),
    arrowHead: {
      enabled: true,
      size: 8,
      angle: Math.PI / 16
    },
    visible: true
  });
```

4.1.3 3D

https://jsfiddle.net/WigandR/q8f2kmyj/

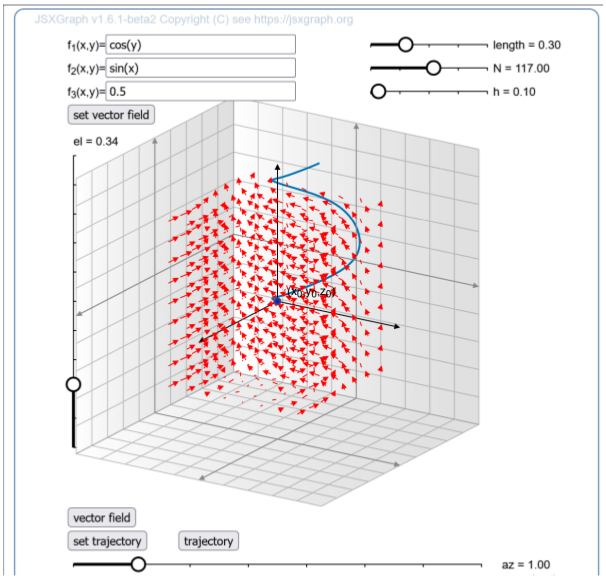

Figure 5: Vector field 3D

In the 3D case each single arrow has be plotted as `line3d`

4.2 Draw the trajectory

JSXGraph provides the Runge-Kutta methods to integrate systems of differential equations `JXG.Math.Numerics.rungeKutta`

Syntax `JXG.Math.Numerics.rungeKutta(butcher, x0, I, N, f)`

Input	value	Realisation
`butcher`	'euler', 'heun', 'rk4'	hard coded
`x0`	Startwert	point in $\mathbb{R}^2, \mathbb{R}^3$
`I`	Intervall [t_0,t_1]	`t_0=0, t_1=N*h`, N,h als slider
`N`	number of steps	slider
`f`	rhs of the ODE	imported from input box (JessieCode)

```
curveC = board.create('curve', [[], []],{strokeWidth:3});
curveC.updateDataArray= function(){
   this.dataX=[];
   this.dataY=[];
   var h = curveh.Value();
   var N = curveN.Value();
   var data = JXG.Math.Numerics.rungeKutta('rk4',
       [point.X(),point.Y()], [0,h*N],N,f);
   for(var i=0; i<curveN.Value(); i++){
      this.dataX[i]=data[i][0];
      this.dataY[i]=data[i][1];
      }
}
```

In the case `vectorfield` some changes have to be done. The vectofield the part of an ODE

$$\dot{\mathbf{y}}(t, \mathbf{x}) = \mathbf{f}(t, \mathbf{x}).$$

There for the function $\mathbf{f}(x, y)$ is rewritten as $\mathbf{f}(x_0, x_1) = \begin{pmatrix} f_x(t, (x_0, x_1)) \\ f_y(t, (x_0, x_1)) \end{pmatrix}$

```
   ftxt = '[' + inputfx.Value() + ',' + inputfy.Value() + ']';
   ftxt = ftxt.replace(/x/g, "x[0]");
   ftxt = ftxt.replace(/y/g, "x[1]");

   // generated function used in Numerics.rungeKutta
   f = board.jc.snippet(ftxt, true, 't,x');
```

With this small adjustment the call to create the curve is the same as stated above.

Update process for `curve3D` (UpdateDataArray) not used yet.

4.3 Vector fields at surfaces and curves

4.3.1 Surfaces

Now come back to the problem stated above.

https://jsfiddle.net/WigandR/dLx2fbus/

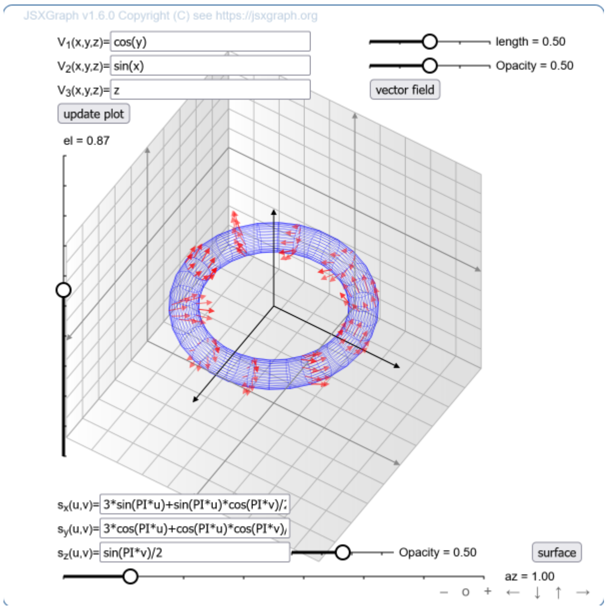

Figure 6: Vector field at a surface

The objects can be (un)hided by the buttons `vector field` and `surface`.

4.3.2 Curve

To get from a surface down to a curve is now only a small change.

https://jsfiddle.net/WigandR/e8btn2p4/

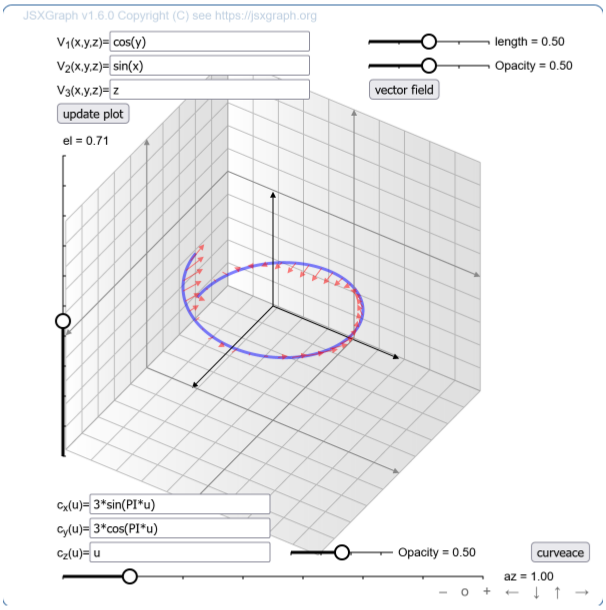

Figure 7: Vector field at a surface

5 Remark

The applets are designed stand alone. All input can be substituted by hard coded definitions or input coming from STACK question variables using the `{#varname#}` syntax.

As last example I will refer a proof of concept, where the vector field depends on a slider value.

https://jsfiddle.net/WigandR/fvatr1wd/

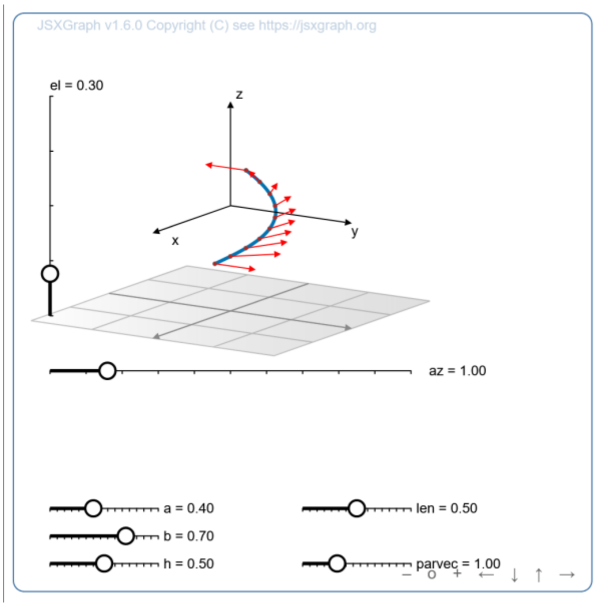

Figure 8: Vector field at curve

6 Helper function

Switch on/off elements in the applet.

```
function toggleList(inputList) {
   for (let iloop = 0; iloop < inputList.length; iloop++) {
      if (JXG.exists(inputList[iloop])) {
         if (inputList[iloop].getAttribute("visible")) {
               inputList[iloop].setAttribute({ visible: false });
         } else {
               inputList[iloop].setAttribute({ visible: true });
               inputList[iloop].show();
         }
         inputList[iloop].update();
      }
   }
}
```

7 Acknowledgement

Thanks to Alfred Wassermann for the discussions and all the improvements inside the JSXGraph core since 1.4.4.

This shown examples were supported by ERASMAUS+ Interactive Digital Assessment in Mathematics (IDIAM). The results of the working package are published at https://idiamath.github.io or https://idiamath.github.io/JSXGraphExamples/JSXGraphExamples.html

Address

Wigand Rathmann

Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU),

Department Mathematik,

wigand.rathmann@fau.de