Workshop: Advanced JSXGraph

Vol. 3

Alfred Wassermann

UNIVERSITÄT BAYREUTH

20-01-2021

Contents

Preliminaries

Include JSXGraph

- JSXGraph skeleton page:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>JSXGraph template</title>
    <meta content="text/html; charset=utf-8" http-equiv="Content-Type">
    <link href="https://cdn.jsdelivr.net/npm/jsxgraph@1.2.1/distrib/
        jsxgraph.css" rel="stylesheet" type="text/css" />
    <script src="https://cdn.jsdelivr.net/npm/jsxgraph@1.2.1/distrib/
        jsxgraphcore.js" type="text/javascript" charset="UTF-8"></script>

    <!-- The next line is optional: MathJax -->
    <script src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-chtml.js"
        id="MathJax-script" async></script>
  </head>
  <body>

  <div id="jxgbox" class="jxgbox" style="width:500px; height:200px;"></div
      >

  <script>
    var board = JXG.JSXGraph.initBoard('jxgbox', {boundingbox: [-5, 2, 5,
        -2]});
  </script>

  </body>
</html>
```

- See JSXGraph handbook (in development): https://ipesek.github.io/jsxgraphbook/

Non-standard scaling of axes

- Update from vol 2, see https://jsxgraph.org/webinar/advanced2.pdf

New features in JSXGraph v1.2.0

New release cycle

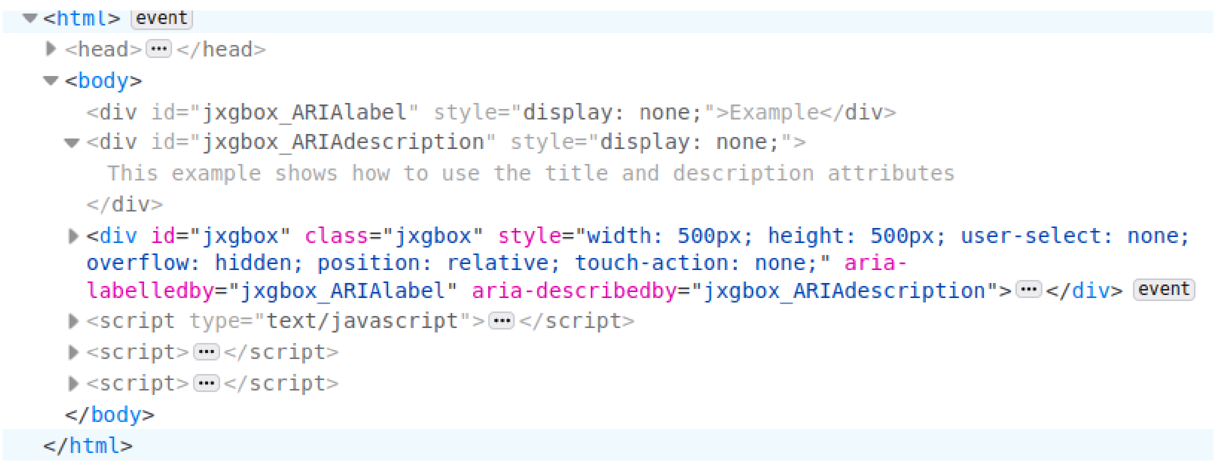
- Starting from v1.2.0, bug fixes will be frequently fixed in the patch releases, increasing the patch release number v1.2.*.
- Today, v1.2.1 has been released.

New attributes `title` and `description` to enhance accessibility

Adding the attributes `title` and `description` to the board attributes will add the ARIA tags `{boardid}_ARIAlabel` and `{boardid}_ARIAdescription` to the HTML DOM in order to enhance accessibility of the construction. Here is the source:

```
const board = JXG.JSXGraph.initBoard('jxgbox', {
    boundingbox: [-5, 5, 5, -5], axis:true,
    title: 'Example',
    description: 'This example shows how to use the title and description
       attributes'
});
```

Here is the DOM:



```
▼<html> event
  ▶<head>…</head>
  ▼<body>
     <div id="jxgbox_ARIAlabel" style="display: none;">Example</div>
    ▼<div id="jxgbox_ARIAdescription" style="display: none;">
       This example shows how to use the title and description attributes
     </div>
    ▶<div id="jxgbox" class="jxgbox" style="width: 500px; height: 500px; user-select: none;
     overflow: hidden; position: relative; touch-action: none;" aria-
     labelledby="jxgbox_ARIAlabel" aria-describedby="jxgbox_ARIAdescription">…</div> event
    ▶<script type="text/javascript">…</script>
    ▶<script>…</script>
    ▶<script>…</script>
   </body>
 </html>
```

Figure 1: DOM screenshot

See https://jsfiddle.net/hq34bvL1/

Further new board attributes

```
const board = JXG.JSXGraph.initBoard('jxgbox', {
    boundingbox: [-5, 5, 5, -5], axis:true,
    drag: { enabled: false},
    showInfobox: false,
    maxFrameRate: 20
});
var p = board.create('point', [1, 1], {showInfobox: `inherit`});

// Enable drag again:
board.attr.drag.enabled = true;
```

See https://jsfiddle.net/82ompdu9/.

`drag`

Similar to `zoom` and `pan`, with the new attribute `drag`, dragging of elements can be enabled (default) or disabled.

`showInfobox`

`showInfobox` is an attribute of points. The value of the new board attribute `showInfobox` will be taken by all points with attribute `showInfobox:'inherit'`, which is the new default value.

maxFrameRate

Maximum frame rate of the board, i.e. maximum number of updates per second *triggered by move events*. Default value is 40.

New ticks attribute `beautifulScientificTickLabels`

```
const board = JXG.JSXGraph.initBoard("jxgbox", {
    boundingbox: [-500000, 500000, 500000, -500000],
    axis: true,
    defaultAxes: {
        x: {
            scalable: true,
            ticks: {
                beautifulScientificTickLabels: true
          },
        },
        y: {
            scalable: true,
            ticks: {
                beautifulScientificTickLabels: true
          },
        }
    },
});
```

- This example uses also the attribute `scalable` which enables horizontal and vertical zooming by dragging the first positive major tick close to its axis.
- See https://jsfiddle.net/493u08jb/

Sensitivity / precision

- There is the board attribute `precision` which sets the sensitivity for all elements, see https://jsxgraph.org/docs/symbols/JXG.Options.html#precision. The precision can be set individually for `mouse`, `touch` and `pen`.
- In v1.2.0 the attribute `precision` has been introduced for all JSXGraph elements with default value `inherit`, see https://jsfiddle.net/8kLrvue1/

```
const board = JXG.JSXGraph.initBoard('jxgbox', {
    boundingbox: [-5, 5, 5, -5], axis:true
});

var line1 = board.create('line', [[-3,2], [3,2]]);
var line2 = board.create('line', [[-3,-2], [3,-2]], {
    precision: {
        mouse: 100,
        pen: 100,
        touch: 100
  }});
```

METAPOST curves

METAPOST is a variant of Donald Knuth's METAFONT system to create fonts for TEX. It outputs PostScript graphics.

METAFONT and METAPOST contain an algorithm by John D. Hobby to produce *good looking* Bezier curves without the need to supply control points. Bezier curves are polynomial curves of degree 3. The Hobby algorithm is the default path algorithm of METAPOST, see e.g. https://bosker.wordpress.com/2013/11/13/beyond-bezier-curves/. Hobby splines are also available in Apple's *pages* software.

- METAPOST in JSXGraph, see https://jsfiddle.net/q0nsad1e/:

```
const board = JXG.JSXGraph.initBoard('jxgbox', {
    boundingbox: [-7, 7, 7, -7], axis:true
});

var po = [],
    attr = {
        size: 5,
        color: 'red'
    },
    controls;

var tension = board.create('slider', [[-3, 6], [3, 6], [0, 1, 20]], {name:
     'tension'});
var curl = board.create('slider', [[-3, 5], [3, 5], [0, 1, 30]], {name: '
    curl A, D'});
var dir = board.create('slider', [[-3, 4], [3, 4], [-180, 0, 180]], {name:
     'direction B'});

po.push(board.create('point', [-3, -3]));
po.push(board.create('point', [0, -3]));
po.push(board.create('point', [4, -5]));
po.push(board.create('point', [6, -2]));

var controls = {
        tension: function() {return tension.Value(); },
        direction: { 1: function() {return dir.Value(); } },
        curl: { 0: function() {return curl.Value(); },
                3: function() {return curl.Value(); }
            },
        isClosed: false
    };

// Plot a metapost curve
var cu = board.create('metapostspline', [po, controls], {
    strokeColor: 'red', strokeWidth: 2
});
```

- As an example, we used METAPOST curves to compute the control points of Bezier curves for the new arrow heads, see https://jsfiddle.net/zxpt9ns1/. This application is a port of https://staff.fnwi.uva.nl/h.vandermeer/pubs/blockarrowmaps.pdf
- Here are the new arrow head types ins JSXGraph (types 4, 5, 6): https://jsfiddle.net/43q96nvc/

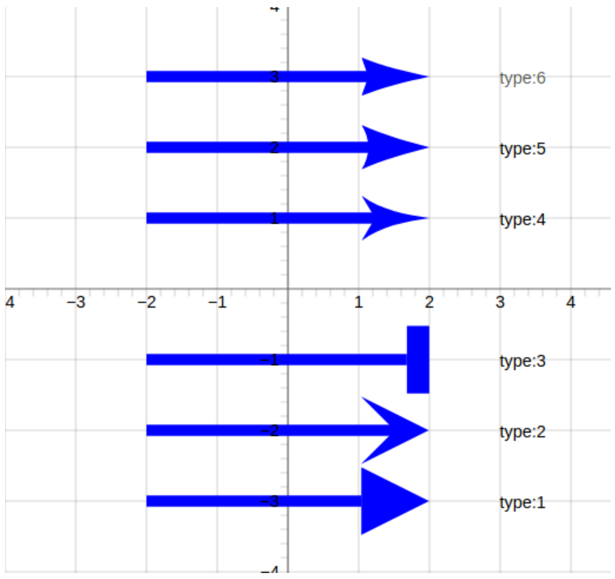


Figure 2: JSXGraph arrow heads

Clipping

Clipping (intersection, union, difference) of curves, circles and polygons has been much improved for so called degenerated cases.

Experimental new features in v1.2.0

Interval arithmetics

```
var I = JXG.Math.IntervalArithmetic;
var i1 = I.Interval(1),
    i2 = I.Interval(2),
    i3;

i3 = I.add(i1, i2);
i3.print();

i1.set(-1, 2);
I.log(i1).print();
```

- See https://jsfiddle.net/z4nhjsv7/
- API documentation is still missing.
- Interval arithmetics will be useful in forthcoming new algorithms for curve plotting and implicit plotting.

New curve plotting algorithm

- The default plotting algorithm in versions v0.99.* up to v1.1.0 is now chosen with

```
var g = board.create('functiongraph', ['log(x)'], {plotVersion: 4});
var g = board.create('functiongraph', [x => Math.log(x) - 3], {plotVersion
    : 2});
```

- The following values are possible:
  - `plotVersion:1`: Very old plotting algorithm - not recommended
  - `plotVersion:2`: default plotting algorithm
  - `plotVersion:3`: unpublished plotting algorithm - not usable
  - `plotVersion:4`: new plotting algorithm, not ready for production - *please give feedback*
- The new plotting algorithm (`plotVersion:4`) uses a novel, very interesting algorithm to detect critical points, and interval arithmetics to determine asymptotics.
- For functions which are highly periodic, `plotVersion:2` is still the better choice.

However: Every plotting algorithm is doomed to fail at one point.

Examples

In the new algorithm, interval arithmetics is only supported for function terms that are supplied in JessieCode syntax, i.e. as string.

Here is a zoo of several functions graph of verying difficulty to plot, see https://jsfiddle.net/z82qucej/28/

```
// JXG.Options.curve.plotVersion = 4;

const board = JXG.JSXGraph.initBoard('jxgbox', {
    boundingbox: [-3, 40, 3, -30],
    axis: true,
    defaultAxes: {
      y: { scalable: true }
    }
});


// Zoo of function graphs
// JavaScript:

// board.create('functiongraph', [x => Math.pow(x, 2 / 3)]);
// board.create('functiongraph', [x => JXG.Math.ratpow(x, 2, 3) + 1]);

// JessieCode:

// board.create('functiongraph', ['3 * x']);
board.create('functiongraph', ['x^5']);
// board.create('functiongraph', ['10*abs(x)']);

// board.create('functiongraph', ['tan(x - PI/2)']);

// board.create('functiongraph', ['log(x)']);
// board.create('functiongraph', ['log(abs(x) - 1)']);
// board.create('functiongraph', ['(x+1)/abs(x+1) * log(abs(x+1))']);
// board.create('functiongraph', ['4 / x']);
// board.create('functiongraph', ['(x + 2)^2 / (x-1) / (x+1)']);

// board.create('functiongraph', ['1 + x * x + 0.0125 * log(abs(1 - 2 * (x
    - 1)))']);

// board.create('functiongraph', ['x * (x - 1 - 0.01) / (x - 1)']);
// board.create('functiongraph', ['x * (x - 1 - 0.000000001) / (x - 1)']);

// board.create('functiongraph', ['8 * x / abs(x)']);

// board.create('functiongraph', ['10 * sin(1 / x)']);
// board.create('functiongraph', ['10 * sin(30 *PI * x)']);
// board.create('functiongraph', ['10 * sin(x) / x']);
// board.create('functiongraph', ['10 * sin(100 * x^2)']);
```

Discussion and suggestion of further topics

- Please, make suggestion for a new element `vectorfield` at https://github.com/jsxgraph/jsxgraph/issues/333
- Our EU project ITEMS will hold a MOOC on *JSXGraph programming for beginners* in April 2021. Stay tuned for details.

Next webinar

The next webinar will be **Wednesday, February 24th, 2021 at 4 pm CET**