ISSN 2942-1373

6

October 8 – 10, 2024

5th International JSXGraph Conference

Book of Abstracts

University of Bayreuth
Center for Mobile Learning with Digital Technology

CMLDT publications

Publisher	University of Bayreuth Center for Mobile Learning with Digital Technology (CMLDT) Universitätsstraße 30 95447 Bayreuth Germany +49 921 55 3266		
Series	CMLDT publications		
ISSN	2942-1373		
Editor	Carsten Miller, carsten.miller@uni-bayreuth.de Alfred Wasserman, alfred.wassermann@uni-bayreuth.de		
License	CMLDT publications © 2023 by University of Bayreuth – Center for Mobile Learning with Digital Technology is licensed under CC BY-ND 4.0. To view a copy of this license, visit http://creativecommons.org/licenses/by-nd/4.0/		
Layout	Carsten Miller, carsten.miller@uni-bayreuth.de		
Internet	https://mobile-learning.uni-bayreuth.de/CMLDTpublications		
Issue	6	**Year of publication**	2025
Title	5th International JSXGraph Conference – Book of Abstracts		

UNIVERSITÄT BAYREUTH

5th International JSXGraph Conference 2024

University of Bayreuth
Center for Mobile Learning with Digital Technology
95440 Bayreuth
Germany

Conference

The 5th International JSXGraph Conference took place from 8th until 10th of October 2024. The online format encouraged fruitful discussion and collaboration among users from all over the world. The schedule respected the location and the timezone of the speakers. The conference was organized by Carsten Miller and Alfred Wassermann from the University of Bayreuth, Germany.

Conference topics

- Usage of JSXGraph
  - for learning / teaching
  - e-Learning environments: moodle, ilias, STACK, ...
  - dynamic visualizations
- Best practices
- Tools
- Presentation of new JSXGraph developments

Website

The abstracts of the talks at the 5th International JSXGraph Conference are also available on the JSXGraph website:

https://jsxgraph.org/conf2024

Videos

Most of the recorded videos of the talks can be found on JSXGraph‘s YouTube Channel:

https://www.youtube.com/@jsxgraph4224

Playlist "5th International JSXGraph Conference"

https://youtube.com/playlist?list=PLr10cPSXxWJfZ6R7vJF3G6yeTfhuPwzJ4&si=PJg1FHan5ugUKUHc

Talks at the 5th International JSXGraph Conference

Carsten Miller, Alfred Wassermann
4 Workshop: JSXGraph for beginners

Kinga Sipos
5 Programming JSXGraph using AI

David Flenner
6 Developing a JSX Graph Workflow

Taha Mzoughi
7 Using JSXGraph in MyOpenMath to Write Introductory Physics Questions

Boris Goldowsky, Leslie Bondaryk, Kirk Swenson
8 Using JSXGraph in a Collaborative Learning Environment

Carsten Miller, Andreas Walter
9 News from sketchometry

Alfred Wassermann
10 Advanced workshop I & II

Andreas Maurischat
11 JSXGraph 3D in MUMIE

Aaron Fenyes
12 Workshop: A hands-on tour of some new 3D elements

Wigand Rathmann
13 Examples using the CAS of JSXGraph

Workshop: JSXGraph for beginners

Carsten Miller
carsten.miller@uni-bayreuth.de
Alfred Wassermann
alfred.wassermann@uni-bayreuth.de
Center for Mobile Learning with Digital Technology
University of Bayreuth
Germany

This workshop will cover the basics of working with JSXGraph. The nice thing about the JSXGraph is that you do not need to be a programmer to use it. In the workshop, we will talk about the tools and how to prepare the working environment. By constructing the examples, we will cover some basic objects as points, lines, circles, intersections, and angles.
Throughout the workshop, we will use the JSXGraph book (https://ipesek.github.io/jsxgraphbook/), which has been created in the European Erasmus+ project ITEMS (https://itemspro.eu) and is freely available on the internet.

Programming JSXGraph using AI

Kinga Sipos
kinga.sipos@unibe.ch
University of Bern
Switzerland

Visualisations play a crucial role in education, particularly in mathematics, by aiding the understanding of complex concepts. Static visualisations help structure and represent knowledge in more concentrated, often simplified ways that cater to different learning styles. Meanwhile, interactive graphs offer students the chance to explore concepts independently, fostering deeper intuition and increasing engagement with the learning material.

In this presentation, I will share my experience using JSXGraph integrated into the ILIAS plugin for STACK, exploring its potential for creating interactive educational tools. With no prior experience in programming JSXGraph, I relied on AI-driven manual prompting through ChatGPT to develop my first exercises. This presentation will highlight the process, challenges, and outcomes of using AI to code JSXGraph visualisations, demonstrating how AI can assist educators in enhancing digital learning experiences.

Developing a JSX Graph Workflow

David Flenner
flennerdr@cofc.edu
College of Charleston
Department of Mathematics
Charleston
SC, USA

I have been developing with JSX Graph since 2017 when I started making interactive visualizations for an online Trigonometry class I was creating at the time. Since then, I have created over a hundred online applications that I use in my teaching daily, and all hosted on GitHub so that anyone who would like to learn the mathematics from them - or how they are programmed - are freely able to. In this talk, I would like to share my experiences in developing JSX applications so that other instructors may feel more comfortable learning how to do the same. My classroom has been transformed by all the apps I use during class to demonstrate difficult topics that lend themselves to a visual explanation, and I would like to help you do the same for yours. I will be demonstrating a few select applications that I have developed, the basics of how they are programmed, and finally how I utilize GitHub to share them. The intended audience is mainly focused upon teachers.

Using JSXGraph in MyOpenMath to Write Introductory Physics Questions

Taha Mzoughi
tmzoughi@gsu.edu
Georgia State University
Georgia, USA

I will give a talk about the use of JSXGraph to develop new introductory physics questions. These questions were developed using MyOpenMath, an online course management and assessment system for mathematics and other quantitative disciplines. MyOpenMath is a nonprofit organization made up of a collaborative community of educators focused on providing algorithmically generated assessment to support the use of free, open textbooks. JSXGraph can be used either by using a JSXGraph derivative provided as a macro or by linking to the web version. These questions are intended for introductory level university or upper level high school physics courses. JSXGraph was used for providing illustrations and interactivity. In the talk, several of the questions will be demonstrated.
The talk is suitable for both school and university educators.

Using JSXGraph in a Collaborative Learning Environment

Boris Goldowsky, Leslie Bondaryk, Kirk Swenson
lbondaryk@concord.org
Concord Consortium, Concord,
MA, USA

Concord Consortium's Collaborative Learning User Environment (CLUE) is a web-based application that allows students to share mathematics work in small groups or across the class either in person or asynchronously. CLUE documents can contain multiple types of "tiles", including text, images, tables, and graphs. JSXGraph underlies the implementation of CLUE's Coordinate Grid tile. This environment is used in research programs to study student acquisition of math concepts. Some unique features of the implementation include

- Real-time collaboration: changes made by one user in a JSXGraph diagram can be watched in real time by other students and teachers.
- History tracking: we record not just the current state of the diagram, but all previous states. Students and teachers can replay changes, and the student's process as well as their final product is visible to their teacher and our researchers.
- Data linking: data can be shared between tiles, so the correspondence between x, y values in a table and geometry objects constructed from these points can be explored.

We will show some examples of how this functionality is used in the classroom, and talk about the benefits that JSXGraph brought to the project, as well as a few challenges that we encountered during the integration.

News from sketchometry

Carsten Miller
carsten.miller@uni-bayreuth.de
Andreas Walter
andreas.walter@uni-bayreuth.de
Center for Mobile Learning with Digital Technology
University of Bayreuth
Germany

sketchometry - in touch with geometry!

At the Center for Mobile Learning with Digital Technology we are not only developing the JavaScript library JSXGraph, but also the gesture-based geometry software sketchometry.
You draw with your finger on your smartphone or tablet. sketchometry converts your sketches into precise geometric constructions that you can modify and move. Mathematics becomes an experimental subject. No ready-made results are presented, learners are encouraged to go on a journey of discovery by themselves. sketchometry is free of charge and can be used both at school and at home.
Last year, sketchometry 2 was officially released: https://start.sketchometry.org.
Since the last conference, however, a lot has happened thanks to the developments in JSXGraph. In our short presentation, we will also briefly discuss a few new features.
We would also be delighted if you could help to translate sketchometry into other languages. Contact us here: https://translate.sketchometry.org/

Advanced workshop I & II

Alfred Wassermann
alfred.wassermann@uni-bayreuth.de
Center for Mobile Learning with Digital Technology
University of Bayreuth
Germany

In two workshops we will cover more advanced topics of JSXGraph programming. The topics will mostly be oriented on changes that happened in JSXGraph since the last conference, that is since version v1.6.2. The plan so far is to cover:

- Progress in 3D
- Using implicit curve plotting
- Update on available statistical methods
- Label positioning for lines, circles and curves
- Showcase the recent overhaul of elements “grid” and “axis”
- Examples for working with complex numbers

JSXGraph 3D in MUMIE

Andreas Maurischat
andreas.maurischat@integral-learning.de
integral-learning
Germany

In a joint project with RWTH Aachen University, we incorporated the usage of JSXGraph 3D in our MUMIE framework. This new development inherits the usual features that MUMIE supports: Easy programming for lecturers in LaTeX-style code; automatic interaction among canvases and between canvases and text; usage as graphical exercises.
In this talk, we present some examples from several variable calculus.

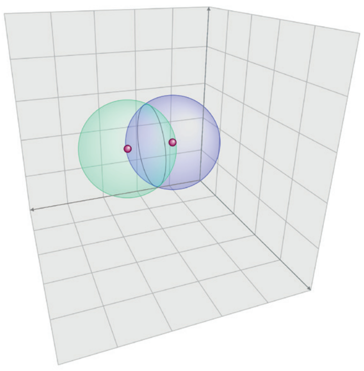
A hands-on tour of some new 3D elements

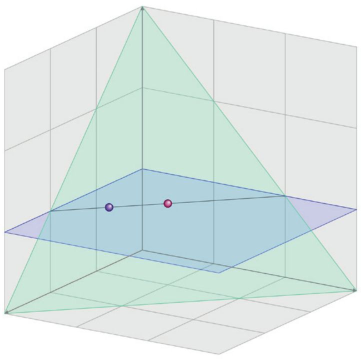
Aaron Fenyes

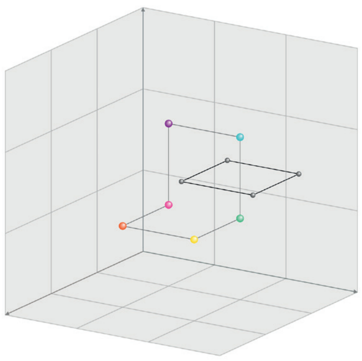
aaron.fenyes@fareycircles.ooo

Over the summer, JSXGraph gained some new 3D elements and capabilities. This short workshop will help you start using them in your constructions. After seeing examples of the new features in action, you'll have time to explore them yourself by building on JavaScript templates that I'll provide.

The new 3D elements include spheres, circles, polygons, and a few gliders and intersections. The new capabilities include "virtual trackball" navigation and depth-ordering for points.







Using CAS of JSXGraph

Wigand Rathmann
Friedrich-Alexander-Universität Erlangen-Nürnberg
Department Mathematik
Erlangen, Germany

1 Abstract

I use JSXGraph a lot in my engineering mathematics classes. I can set up a board for special purposes, perhaps by using a CAS like Maxima. This often follows the typical cycle of developing and debugging a JSXGraph applet:

🤔😕😫🤔😡😮🤦😊😅😎

Thanks to getDigital

When an applet works as expected, you are usually very happy. At this point the loop of improvements and the search for tools to realise the ideas (and the development and debugging phases) starts.

My idea is often to create flexible JSXGraph applets, where the user can insert information directly into the applet. At this point I look for solutions that run entirely within JSXGraph. Mostly I use numerical approaches, but sometimes it is better or just nicer to have an analytical approach.

In calculus we deal a lot with functions and their derivatives. When thinking about tangents on a graph, these are three lines for a fixed function to have JSXGraph board.

```
var graph = board.create('functiongraph', ["sin(x)", -10, 10], {
     strokeColor: '#00ff00'});
var p1 = board.create('glider', [0,0,graph], {style:6, name:'P'
   });
var t = board.create('tangent', [p1]);
```

To be more flexible, one can use an input box to enter and change the function term and e.g. show tangents. The next step could be to demonstrate a Taylor polynomial $T_{f,n}$ of degree n for a chosen function f (provided by the user) at point $x0$:

$$T_{f,n}(x, x_0) = \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(x_0)(x - x_0)^n.$$

.

And now I need “good” derivatives. JSXGraph comes with some computer algebra algorithms. The functionality is provided to the user via JessieCode, a “scripting language designed to provide an interface to JSXGraph”. Two functions are used to handle function expressions: `jc.snippet` to generate a JavaScript function from the function string and `jc.manipulate` to calculate derivatives.

In the same way, Maxima and JSXGraph can be linked in STACK questions. This allows you to develop and test JSXGraph applets before using them in a STACK question. The derivatives of a function can be calculated in Maxima, but some symbols have to be replaced. For example, the symbols for π or e are different in Maxima and JSXGraph. At this point the powerful member function `replace` of strings helps and supports regular expressions.

In this talk I will show how to pass a function expression from an input field to the JSXGraph objects and how I used it to

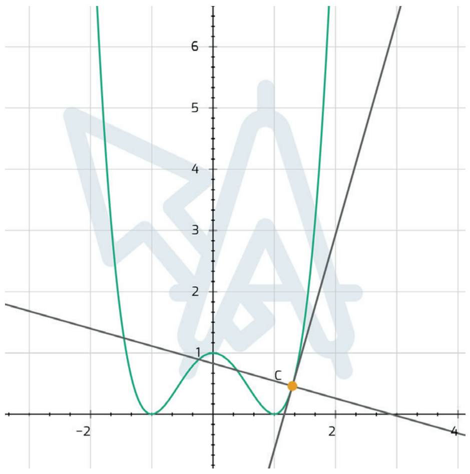
- Construct the osculating circle on a graph,
- plot the Taylor polynomial on a graph,
- construct the curl $\nabla \times V$ of a given vector field $V : \mathbb{R}^3 \to \mathbb{R}^3$,
- give an impression of atomic orbitals using the spherical harmonic function Y_l^m of degree l and order m“.

2 Examples

2.1 Osculating circle on a graph

Given $f \in C^2((a, b))$. The graph we interpret as a curve k in $\mathbf{R}^2$ with $k(t) = \begin{pmatrix} t \\ f(t) \end{pmatrix}$ and $t \in [\hat{a}, \hat{b}] \subset (a, b)$. The osculation circe at a point $k(t_0)$ touches the graph of f with the same curvature.

To construct the centroid of the circle we need the tangent at $k(t_0) = \begin{pmatrix} t_0 \\ f(t_0) \end{pmatrix}$ given by $\dot{k}(t_0) = \begin{pmatrix} 1 \\ \dot{f}(t_0) \end{pmatrix}$. The normal at the point is given by $n_k(t) = \begin{pmatrix} -\dot{f}(t) \\ 1 \end{pmatrix}$.



Tangent and normal vector

That means, the centroid of the osculation circle at $k(t_0)$ is lying on the line

$$k(t_0) + \lambda n_k(t_0), \lambda \in \mathbb{R}$$

In a convex part of function the osculationg circle is above the graph, hence λ has to be positive. Form differential geometry we the curvature is given by

$$\kappa = \frac{f''(x)}{(1 + f'(x)^2)^{\frac{3}{2}}}.$$

By the sign of f'' we also get the right direction, where the circle has to be placed.

To realize the input, to compute the derivatives and to generate the JavaScript function used to draw everything, it needs a few lines only:

```
        strf = inputfx.Value();
        strDf = b1.jc.manipulate("D(" + strf + ",x);");
        strD2f = b1.jc.manipulate("D(" + strDf + ",x);");
        f = b1.jc.snippet(strf, true, 'x');
        Df = b1.jc.snippet(strDf, true, 'x');
        D2f = b1.jc.snippet(strD2f, true, 'x');
```

2.1.1 Summary

Turn a string `strf` into a JavaScript function `fun`: `fun = b1.jc.snippet(strf, true, 'x');`

Differentiate a function the sytax is: `D(<functionstring>,<variabename>[,order])`, e.g. `D(sin(t),t,2)` will return $\frac{\mathrm{d}^2}{\mathrm{d}t^2}\sin(t)$.

The obtain the derviative of a function stored in `strf=x^3+sin(x)` one has to combine it like `D(`+strf+`,x,2)` Get the `n`th derivative of `strf`: `strDf = b1.jc.manipulate("D("+ strf + ",x,n);");`

Turn it into a funciton using: `funDx = b1.jc.snippet(strDf, true, 'x');`

2.2 Curl of a vector field

In vector analysis the curl is an important concept. The definition is quit simple:

$V : \mathbb{R}^3 \to \mathbb{R}^3$ is at least on time continous partial differentiable.

$$\operatorname{curl} V(\mathbf{x}) = \nabla \times V(\mathbf{x}) = \begin{pmatrix} \frac{\partial}{\partial x_2} V_3(\mathbf{x}) - \frac{\partial}{\partial x_3} V_2(\mathbf{x}) \\ \frac{\partial}{\partial x_3} V_1(\mathbf{x}) - \frac{\partial}{\partial x_1} V_3(\mathbf{x}) \\ \frac{\partial}{\partial x_1} V_2(\mathbf{x}) - \frac{\partial}{\partial x_2} V_1(\mathbf{x}) \end{pmatrix}$$

or in a short form using the total antisymmetric unit tensor (I think physists will like it)

$$\nabla \times V = \epsilon_{ijk} V_{k,j} \mathbf{e}_i .$$

To realize this as an interactive applet, we can extend the idea of the last example. We need three input boxes for the vector field components. The next step is, to implement the curl operator.

```
// input
fx = board.jc.snippet(inputfx.Value(), true, 'x,y,z');
fy = board.jc.snippet(inputfy.Value(), true, 'x,y,z');
fz = board.jc.snippet(inputfz.Value(), true, 'x,y,z');
// partial derivatives
let Dfx2 = board.jc.manipulate("D(" + inputfx.Value() + ",y);");
let Dfx3 = board.jc.manipulate("D(" + inputfx.Value() + ",z);");
let Dfy1 = board.jc.manipulate("D(" + inputfy.Value() + ",x);");
let Dfy3 = board.jc.manipulate("D(" + inputfy.Value() + ",z);");
let Dfz1 = board.jc.manipulate("D(" + inputfz.Value() + ",x);");
let Dfz2 = board.jc.manipulate("D(" + inputfz.Value() + ",y);");
// generate strings for compontents of curl V
let curlx = Dfz2 + "-(" + Dfy3 + ")";
let curly = Dfx3 + "-(" + Dfz1 + ")";
let curlz = Dfy1 + "-(" + Dfx2 + ")";

Fcurlx = board.jc.snippet(curlx, true, 'x,y,z');
Fcurly = board.jc.snippet(curly, true, 'x,y,z');
Fcurlz = board.jc.snippet(curlz, true, 'x,y,z');
```

At this point the vectorfield V is available as `[fx,fy,fz]` and $\nabla \times V$ as `[Fcurlx,Fcurly,Fculz]`.

I am not sure, if this visualisation will help students to improve the understanding of the curl.

2.3 Taylor polynomial

To demonstrate Taylor polynomials several applets will be available. Usually the function ist fix it is quit nice to implement, see Taylor polynomial of sine.

Let's resume the ingredients: We need a function at least $n+1$ times continuous differentiable.

Then we can write

$$T_{f,n}(x,x0) = \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(x_0)(x-x_0)^n.$$

In JSXGraph the function f we get by

```
ftxt = inputfx.Value();
funf = board.jc.snippet(ftxt, true, 'x');
```

$f^{(k)}$ we get recursively from f using JessiCode and we could store it in an array

```
fdx[0]=ftxt;
fdx.push(board.jc.manipulate('D('+fdx[i-1]+',x)'));
```

Rewrite the sum as a Horner scheme to reduce the computational load:

$$T_{f,n}(x,x0) = \left(\cdots\left(\left(f^{(n)}(x_0)\frac{1}{n}(x-x_0) + f^{(n-1)}(x_0)\right)\frac{1}{n-1}(x-x_0)\right.\right.$$
$$\left.\left. + f^{n-2}(x_0)\right)\cdots + f^{(1)}(x_0)\right)(x-x_0) + f(x_0).$$

In terms of JSXGraph we need the derivatives a point `x_0.X()` on the graph. The Taylor polynomial can now be written in a for loop. Therefor I generated an array containing all derivatives up to order n_max. Thus the evaluation of $f^{(k)}(x_0)$ can be done inside the function graph object of the Taylor polynomial.

```
taylorpol = b1.create('functiongraph', [
        function (t) {
          var h = (t - x0.X());
          val = funf[s.Value()](x0.X());
          for (let i = s.Value() - 1; i >= 0; i--) {
            val = val * h / (i+1) + funf[i](x0.X());
          }
          return val;
        },
        -10, 10],
        { strokeColor: "#bb0000" });
```

2.3.1 Note

At this point I have to note, that JSXGraph is not a CAS. So you may reach some limitations, if your function term is a bit more complex. E.g. for polynomials on should not use the factorized form.

2.4 Orbits of H atoms

This example was the initial idea to think about CAS in JSXGraph. To plot the orbitals the spherical harmonics were used. These functions, denoted by $Y_{l,m}$ are deduced from Legendre polynomials.

I will only state the formulas needed to plot the orbitals. For a more detailed description one can start at the Wikipedia page and some dedicated literature.

2.4.1 Legendre Polynomials P_l

One way to obtain the spherical harmonics is starting with Rodrigues' formula.

$$P_l(x) := \begin{cases} \frac{1}{2^l l!}\frac{\mathrm{d}^l}{\mathrm{d}x^l}(x^2-1)^l, & l \geqslant 1, \\ 1, & l = 0. \end{cases}$$

In JessiCode terms it is

```
basePol = '(x^2-1)';
P1 = basePol + '^' + lVal.Value();
DP1 = boardinput.jc.manipulate("D(" + P1 + ",x," + lVal.Value()
    + ");");
```

But in the current version of `manipulate` this is to general. (Please remember: JSXGraph is a visualization tool and not a CAS like Maxima!).

So I used the list of the first polynomials

$$\begin{aligned} P_1(x) &= x \\ P_2(x) &= (3x^2-1)/2 \\ P_3(x) &= (5x^3-3x)/2 \\ P_2(x) &= (3x^2-1)/2 \\ P_4(x) &= (35x^4-30x^2+3)/8 \\ P_5(x) &= (63x^5-70x^3+15x)/8 \end{aligned}$$

2.4.2 Spherical harmonics

The Spherical harmonics are now defined by $(m \leqslant l)$

$$Y_{l,m}(x) := \begin{cases} (1-x^2)^{\frac{1}{2}m} \dfrac{\mathrm{d}^m}{\mathrm{d}x^m} P_l(x), & m \geqslant 1, \\ P_l(x), & m = 0. \end{cases}$$

The functions have to be normalized by using $Ny = (2l+1)\dfrac{(l-m)!}{(l+m)!}$.

To plot the orbitals only the absolute value of the real part is used. The general approach using the complex domain is dropped for vitalization.

To plot the orbitals $Y_{l,m}$ is read as the radius depending of the point of the unit sphere. To plot the function one needs to substitute $x = \cos(\theta)$, this done in the call of the parametric surface directly:

```
view.create('parametricsurface3d', [
                 (u, v) => Ylm2Fun(Math.cos(v)) * Math.sin(v) *
                    Math.cos(u),
                 (u, v) => Ylm2Fun(Math.cos(v)) * Math.sin(v) *
                    Math.sin(u),
                 (u, v) => Ylm2Fun(Math.cos(v)) * Math.cos(v),
                 () => [phi0.Value(), 2 * Math.PI],
                 [0, Math.PI]
             ])
```

The implementation is based on the work of Micheal Komma.

Komma, Michael: Moderne Physik mit Maple. International Thomson Publishing (1996), Website, Kapitel 5.2.4 H-Atome.

H-Orbitale: http://www.mikomma.de/fh/embuchhtm/hydrogen_10.html

2.5 Examples online

The examples are published at wrathmann.github.io.

- Osculating circle on a graph
- Taylor polynomial
- Curl of a vector field
- Orbitals of H atom