Automatic calculation of plane loci using Groebner bases and integration into a Dynamic Geometry System

Michael Gerhäuser, Alfred Wassermann

ord.createElement('slider' July 24, 2010 rd. createElement ('slider', [[1, notion (x) (return Math. sin (x) brd.createElement('slider' plot = brd.createElement('functiongraph', os = brd.createElement('riemannsum unction(){ return s.Value unction(){return a.Value

nction(){return b.Value

Overview

JSXGraph - A short overview

Computing plane loci using Groebner bases

Implementing this algorithm in JSXGraph

Optimizations

Examples



What is JSXGraph?

- ► A library implemented in JavaScript
- ▶ Runs in recent versions of all major browsers
- No plugins required
- ► LGPL-Licensed

Main features

- Dynamic Geometry
- Interactive function plotting
- Turtle Graphics
- Charts



unction() (return s.Value

Unction() (return a.Value

Unction() (return b.Value)

Supported Hardware

- ► PC (Windows, Linux, Mac)
- Mobile phones
- ▶ "Touchpads" like the Apple iPod and iPad
- Basically everything which runs at least one of the supported browsers

 Brd. createElement ('slider')

plot = brd.createElement('slider', [[1,
plot = brd.createElement('functiong.

os = brd.createElement('functiong.
function() { return s.Value | UNIVE BAYRE

bayes

unction() { return b.Value | UNIVE BAYRE

bayes

unction() { return b.Value | UNIVE BAYRE

bayes

unction() { return b.Value | UNIVE BAYRE

unction() { return b.

Supported Browsers

- ► Firefox
- Chrome/Chromium
- Safari
- ▶ Internet Explorer
- Opera

s = brd.createElement('slider', [[1,3],
f = function(x) { return Math.sin(x); }
s = brd.createElement('slider', [[1,1], [
plot = brd.createElement('functiongraph')
f = function(x) { return Math.sin(x); }
s = brd.createElement('functiongraph')
function() { return s.Value function() { return a.Value function() { return b.Value function()

Example/Input

```
<link rel="stylesheet" type="text/css" href="css/jsxgraph.css" />
<script type="text/javascript" src="is/isxgraphcore.is">>/script>
[...]
<div id="jxgbox" class="jxgbox" style="width:500px; height:500px;">//div>
<script type="text/javascript">
 /* <![CDATA[ */
  board = JXG.JSXGraph.initBoard('jxgbox', {boundingbox: [-2, 20, 20, -2], axis:}
        true, grid: false, keepaspectratio: true, showcopyright: false });
  p3 = board.create('point', [8, 3]);
  p4 = board.create('point', [8, 8]);
  c1 = board.create('circle', [p4, 4]);
  p6 = board.create('glider', [0, 0, c1], {name: 'D'});
  g = board.create('line', [p3, p6]);
  c2 = board.create('circle', [p6, 3]);
  p14_1 = board.create('intersection', [c2,g,0], {name: 'T'});
  /* 11> */
</script>
                                                         return Math.sin(x);
```

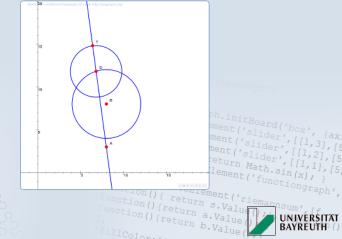
os = brd.createElement('functiongraph',
function()(return s.Value
function() (return a.Value
function() (return b.Value
fillColor:'#fffff

JSXGraph Example/Output Element('slider',[[1,2], lement('slider',[[1,1],[5, teElement('functiongraph', unction() { return s.Value() createElement('riemannsum', [f unction(){return a.Value unction() (return b. Value(

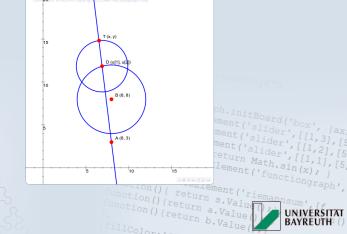
Computing plane loci using Groebner bases (in a nutshell)

```
## Septiment of the state of th
```

Given a set of free and dependent points,



we first choose a coordinate system,



- translate geometric constraints into an algebraic form,
 - $(u[1] 8)^2 + (u[2] 8)^2 16 = 0$
 - $(x u[1])^2 + (y u[2])^2 9 = 0$
 - 3x 3u[1] + yu[1] 8y + 8u[2] xu[2] = 0

brd.createElement('slider', os = brd.createElement('riemannsum'

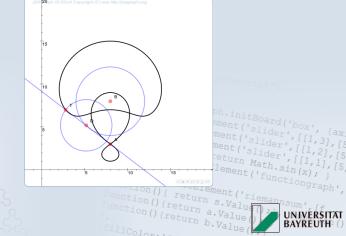
unction() (return s.Valu unction(){return a.Value nction()(return b.Value)

- calculate the Gröbner basis of the given ideal,
 - $x^6 + 3x^4y^2 + 3x^2y^4 + y^6 48x^5 38x^4y 96x^3y^2 76x^2y^3$ $48xv^4 - 38v^5 + 1047x^4 + 1216x^3v + 1774x^2v^2 + 1216xv^3 +$ $727y^4 - 13024x^3 - 16596x^2y - 16096xy^2 - 8404y^3 + 97395x^2 +$ $109888xy + 63535y^2 - 415536x - 300806y + 790009 = 0$

eateElement('slider' os = brd.createElement('riemannsum unction() { return s.Value()

unction(){return a.Value nction(){return b.Value(

and finally plot the calculated implicit equation.



Implementing this algorithm in JSXGraph

Implementing this algorithm in JSXGraph

Problems

- ▶ No JavaScript implementation of any Gröbner basis algorithm
- Can't use C-libraries directly in JavaScript
- No implicit plotting in JSXGraph by now



Implementing this algorithm in JSXGraph

Our solution

XMLHttpRequest/AJAX to

```
s = brd.createElement('slider', [a], a = brd.createElement('functiongraph function() { return s. Value function() { return s. Value function() { return b. Value() } UNIVERSITA BAYREUTH
```

Optimizations

brd = JXG. JSXGraph.initBoard('box', (as a brd.createElement('slider', [[1,3], f f f f.]) brd.createElement('slider', [[1,2], f f f.]) f. [[1,2], f f f.]] f. [[1,2], f f.]] f

Examples