

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

Michael Gerhäuser, Alfred Wassermann

July 24, 2010



UNIVERSITÄT  
BAYREUTH

# Overview

JSXGraph - A short overview

Computing plane loci using Gröbner bases

Implementing this algorithm in JSXGraph

Optimizations

Examples

```
</script>
</html>
</javascript>

var brd = JSXGraph.initBoard('box', {ax:
var s = brd.createElement('slider', [[1,3], [5
var a = brd.createElement('slider', [[1,2], [5
var b = brd.createElement('slider', [[1,1], [5
var f = function(x){ return Math.sin(x); }
var plot = brd.createElement('functiongraph',

var os = brd.createElement('riemannsum', [f
function(){ return s.Value(); }
function(){ return a.Value(); }
function(){ return b.Value(); }
},
{fillColor: '#ffff00'}
```



UNIVERSITÄT  
BAYREUTH

# JSXGraph

```
<script src="http://www.jsxgraph.org/javascript">  
// JSXG  
  
var brd = JSXGraph.initBoard('box', {ax:  
var s = brd.createElement('slider', [[1,3],[5  
var a = brd.createElement('slider', [[1,2],[5  
var b = brd.createElement('slider', [[1,1],[5  
var f = function(x){ return Math.sin(x); }  
var plot = brd.createElement('functiongraph',  
  
var os = brd.createElement('riemannsum', {f:  
function(){ return s.Value(); }  
function(){return a.Value(); }  
function(){return b.Value(); }  
},  
{fillColor:'#ffff00',
```



UNIVERSITÄT  
BAYREUTH

# JSXGraph

## 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

```
<script src="/js/javascript">  
var brd = JSXGraph.initBoard('box', {ax:  
var s = brd.createElement('slider', [[1,3], [5  
var a = brd.createElement('slider', [[1,2], [5  
var b = brd.createElement('slider', [[1,1], [5  
var f = function(x) { return Math.sin(x); }  
var plot = brd.createElement('functiongraph',  
var os = brd.createElement('riemannsum', {f:  
function() { return s.Value(); }  
function() { return a.Value(); }  
function() { return b.Value(); }  
},  
{fillColor: '#ffff00';
```



## Supported Hardware

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

```
<script type="text/javascript">  
var s = JSXGraph.initBoard('box', {ax:  
var a = brd.createElement('slider', [[1,3],[5  
var b = brd.createElement('slider', [[1,2],[5  
var f = function(x){ return Math.sin(x); }  
var plot = brd.createElement('functiongraph',  
var os = brd.createElement('riemannsum', {f:  
function(){ return s.Value(); }  
function(){ return a.Value(); }  
function(){ return b.Value(); }  
},  
{fillColor:'#ffff00';
```



## Supported Browsers

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

```
<script src="/javascript">  
var brd = JSXGraph.initBoard('box', {ax:  
var s = brd.createElement('slider', [[1,3],[5  
var a = brd.createElement('slider', [[1,2],[5  
var b = brd.createElement('slider', [[1,1],[5  
var f = function(x){ return Math.sin(x); }  
var plot = brd.createElement('functiongraph',  
var os = brd.createElement('riemannsum', {f  
function(){ return s.Value(); }  
function(){return a.Value(); }  
function(){return b.Value(); }  
},  
{fillColor:'#ffff00'
```

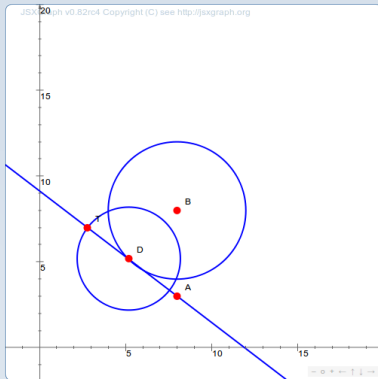


## Example/Input

```
<link rel="stylesheet" type="text/css" href="css/jsxgraph.css" />
<script type="text/javascript" src="js/jsxgraphcore.js"></script>
...
<div id="jxgbox" class="jxgbox" style="width:500px; height:500px;"></div>
<script type="text/javascript">
    board = JSXGraph.initBoard('jxgbox', {boundingbox: [-2, 20, 20, -2], axis:
        true, grid: false, keepaspectratio: true});
    A = board.create('point', [8, 3]);
    B = board.create('point', [8, 8]);
    c1 = board.create('circle', [B, 4]);
    D = board.create('glider', [0, 0, c1], {name: 'D'});
    g = board.create('line', [A, D]);
    c2 = board.create('circle', [D, 3]);
    T = board.create('intersection', [c2,g,0], {name: 'T'});
</script>
```



## Example/Output





## Supported file formats

- ▶ GEONE<sub>x</sub>T
- ▶ GeoGebra
- ▶ Intergeo
- ▶ Cinderella (small feature subset)

```
<script src="/js/javascript">  
var brd = JSXGraph.initBoard('box', {ax:  
  var a = brd.createElement('slider', [[1,3], [5  
  var b = brd.createElement('slider', [[1,2], [5  
  var f = function(x){ return Math.sin(x); }  
  var plot = brd.createElement('functiongraph',  
  var os = brd.createElement('riemannsum', {f:  
    function(){ return s.Value(); }  
    function(){ return a.Value(); }  
    function(){ return b.Value(); }  
  },  
  {fillColor: '#ffff00';
```



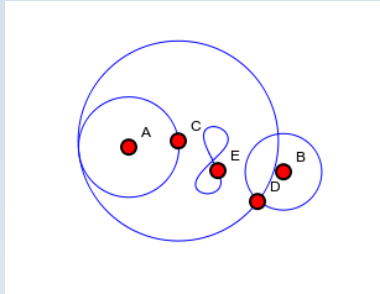
## Example/Input

```
<link rel="stylesheet" type="text/css" href="css/jsxgraph.css" />
<script type="text/javascript" src="js/jsxgraphcore.js"></script>
<script type="text/javascript" src="js/CinderellaReader.js"></script>
...
<div id="jxgbox" class="jxgbox" style="width:500px; height:500px;"></div>
<script type="text/javascript">
    board = JSXGraph.loadBoardFromFile('jxgbox', 'watt.cdy', 'cinderella');

    function computeLocus() {
        board.create('locus', [JXG.getRef('E')]);
    }
</script>
```



## Example/Output



```
graph.initBoard('box', {ax:  
  Element('slider', [[1,3],[5  
  Element('slider', [[1,2],[5  
  Element('slider', [[1,1],[5,  
  plot = brd.createElement('functiongraph',  
  cos = brd.createElement('riemannsum', {f  
  function(){ return s.Value(); }  
  function(){return a.Value(); }  
  function(){return b.Value(); }  
  },  
  {fillColor: '#ffff00',
```



## Computing plane loci using Gröbner bases<sup>1</sup> (in a nutshell)

---

<sup>1</sup>Recio & Vélez 1999

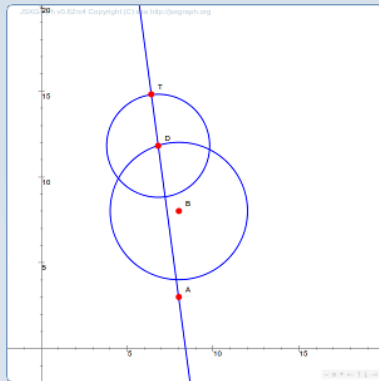
Botana & Valcarce 2002

Botana, Abánades & Escribano 2007



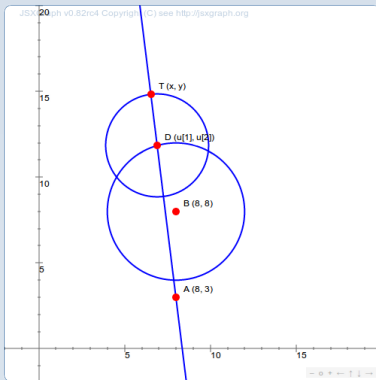
# Computing plane loci using Gröbner bases

- Given a set of free and dependent points,



# Computing plane loci using Gröbner bases

- ▶ we first choose a coordinate system,



# Computing plane loci using Gröbner bases

- ▶ translate geometric constraints into an algebraic form,

- ▶  $(u[1] - 8)^2 + (u[2] - 8)^2 - 16 = 0$

- ▶  $(x - u[1])^2 + (y - u[2])^2 - 9 = 0$

- ▶  $3x - 3u[1] + yu[1] - 8y + 8u[2] - xu[2] = 0$



# Computing plane loci using Gröbner bases

- ▶ calculate the Gröbner basis of the given ideal,

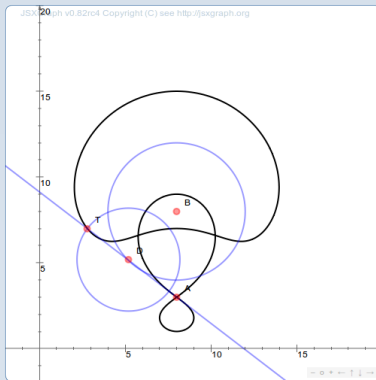
- ▶ 
$$x^6 + 3x^4y^2 + 3x^2y^4 + y^6 - 48x^5 - 38x^4y - 96x^3y^2 - 76x^2y^3 - 48xy^4 - 38y^5 + 1047x^4 + 1216x^3y + 1774x^2y^2 + 1216xy^3 + 727y^4 - 13024x^3 - 16596x^2y - 16096xy^2 - 8404y^3 + 97395x^2 + 109888xy + 63535y^2 - 415536x - 300806y + 790009 = 0$$





# Computing plane loci using Gröbner bases

- ▶ and finally plot the variety generated by the ideal.



## Implementing this algorithm in JSXGraph

```
<script type="text/javascript">  
var brd = JSXGraph.initBoard('box', {ax:  
var s = brd.createElement('slider', [[1,3],[5  
var a = brd.createElement('slider', [[1,2],[5  
var b = brd.createElement('slider', [[1,1],[5  
var f = function(x){ return Math.sin(x); }  
var plot = brd.createElement('functiongraph',  
var os = brd.createElement('riemannsum', {f:  
function(){ return s.Value(); }  
function(){return a.Value(); }  
function(){return b.Value(); }  
},  
{fillColor:'#ffff00'
```



# Implementation

## Problems

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

```
</script>  
</html>  
  
var s = JSXGraph.initBoard('box', {ax:  
var a = brd.createElement('slider', [[1,3],[5  
var b = brd.createElement('slider', [[1,2],[5  
var f = function(x){ return Math.sin(x); }  
var plot = brd.createElement('functiongraph',  
  
var os = brd.createElement('riemannsum', {f:  
function(){ return s.Value(); }  
function(){ return a.Value(); }  
function(){ return b.Value(); }  
},  
{fillColor:'#ffff00'
```



# Implementation

## AJAX

- ▶ Transfer data (a)synchronously via HTTP with JavaScript

## This enables us to

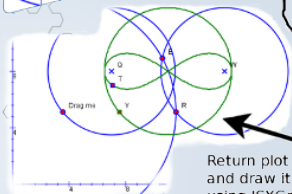
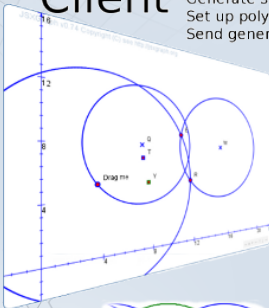
- ▶ use a computer algebra system on a (web) server for the expensive Gröbner basis calculations
- ▶ use a plotting tool/library for implicit plotting



# Implementation

## Client

Generate symbolic coordinates for free and dependent points.  
Set up polynomials for the dependent ones describing their loci.  
Send generated data to webserver for further calculations.



Return plot data  
and draw it  
using JSXGraph.

AJAX



## Server

Several software packages are used server side:



python

is used to retrieve the data  
and pass it on to

CoCoA handles the symbolic  
algebra stuff and returns a  
set of polynomials which are  
plotted with the python library



matplotlib

Finally the locus curve is extracted as a list of  
coordinates from the plots and is sent back to  
JSXGraph where the data is used to plot the  
locus directly in the geometric construction.



UNIVERSITÄT  
BAYREUTH

# Implementation

## Example/Input

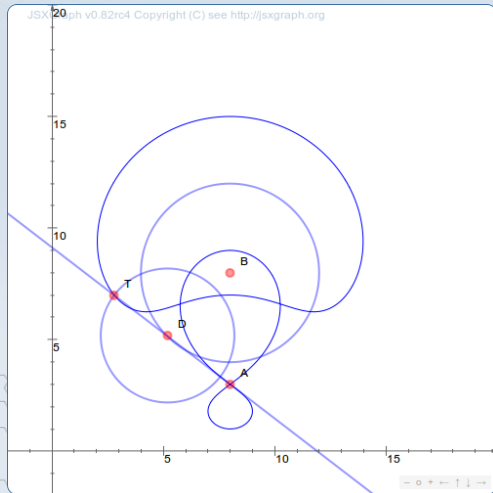
```
<link rel="stylesheet" type="text/css" href="css/jsxgraph.css" />
<script type="text/javascript" src="js/jsxgraphcore.js"></script>
...
<div id="jxgbox" class="jxgbox" style="width:500px; height:500px;"></div>
<script type="text/javascript">
  board = JXG.JSXGraph.initBoard('jxgbox', {boundingbox: [-2, 20, 20, -2], axis:
    true, grid: false, keepaspectratio: true});
  A = board.create('point', [8, 3]);
  B = board.create('point', [8, 8]);
  c1 = board.create('circle', [B, 4]);
  D = board.create('glider', [0, 0, c1], {name: 'D'});
  g = board.create('line', [A, D]);
  c2 = board.create('circle', [D, 3]);
  T = board.create('intersection', [c2,g,0], {name: 'T'});

  locus = board.create('locus', [T]);
</script>
```



# Implementation

## Example/Output



# Implementation

## Ready-to-use elements

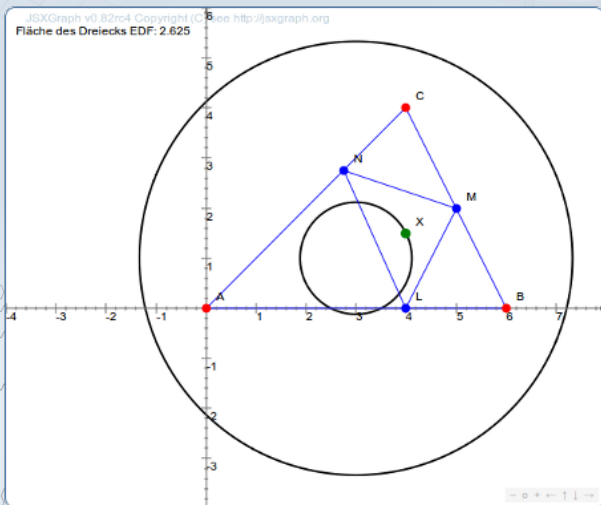
- ▶ Glider on circle and line
- ▶ Intersection points (circle/circle, circle/line, line/line)
- ▶ Midpoint
- ▶ Parallel line and point
- ▶ Perpendicular line and point
- ▶ Circumcircle and circumcenter





# Implementation

Easy to extend



UNIVERSITÄT  
BAYREUTH

# Implementation

```
<link rel="stylesheet" type="text/css" href="css/jsxgraph.css" />
<script type="text/javascript" src="js/jsxgraphcore.js"></script>
<script type="text/javascript" src="js/Triangle.js"></script>
...
<div id="jxgbox" class="jxgbox" style="width:500px; height:500px;"></div>
<script type="text/javascript">
    board = JXG.JSXGraph.initBoard('jxgbox', {boundingbox:[-4, 6, 8, -4], axis:
        true, grid: false, keepaspectratio: true});
    A = board.create('point', [0, 0]);
    B = board.create('point', [6, 0]);
    C = board.create('point', [4, 4]);

    t1 = board.create('triangle', [A, B, C], {strokeWidth: '1px'});

    X = board.create('point', [4, 1.5], {name:"X"});

    L = board.create('perpendicularpoint', [X, t1.c]);
    M = board.create('perpendicularpoint', [X, t1.a]);
    N = board.create('perpendicularpoint', [X, t1.b]);

    t2 = board.create('triangle', [L, M, N], {strokeWidth: '1px'});
```



# Implementation

```
...

X.ancestors[L.id] = L;
X.ancestors[M.id] = M;
X.ancestors[N.id] = N;
X.ancestors[A.id] = A;
X.ancestors[B.id] = B;
X.ancestors[C.id] = C;

X.generatePolynomial = function () {
  var as16 = getTriangleArea(L, M, N),
  as = '((( '+M.symbolic.x+')-('+N.symbolic.x+')^2+(( '+M.symbolic.y+')-('+N.
    symbolic.y+')^2)',
  bs = '((( '+L.symbolic.x+')-('+N.symbolic.x+')^2+(( '+L.symbolic.y+')-('+N.
    symbolic.y+')^2)',
  cs = '((( '+M.symbolic.x+')-('+L.symbolic.x+')^2+(( '+M.symbolic.y+')-('+L.
    symbolic.y+')^2)',

  return [ '4*'+as+'*'+cs+'-('+as+'+'+cs+'-'+bs+')*('+as+'+'+cs+'-'+bs+')-('+
    as16+')' ];
};

locus = board.create('locus', [X], {strokeColor: 'red'});
</script>
```



# Implementation

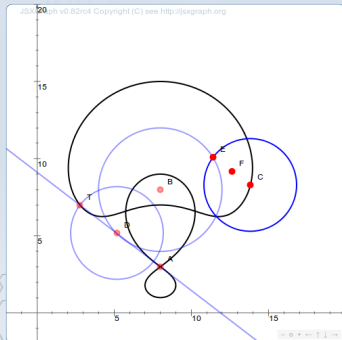
Re-using locus data: Discovered loci can be

- ▶ intersected with circles, lines, other curves, ...
- ▶ used as a base object for gliding points
- ▶ used for the discovery of other loci

```
var s = JXG.JSXGraph.initBoard('box', {ax:  
var a = brd.createElement('slider', [[1,3], [5  
var b = brd.createElement('slider', [[1,2], [5  
var f = function(x) { return Math.sin(x); }  
var plot = brd.createElement('functiongraph',  
var os = brd.createElement('riemannsum', {f:  
function() { return s.Value(); }  
function() { return a.Value(); }  
function() { return b.Value(); }  
},  
{fillColor: '#ffff00',
```



# Implementation



```
C = board.create('glider', [locus]);
```

```
c2 = board.create('circle', [C, 3]);
```

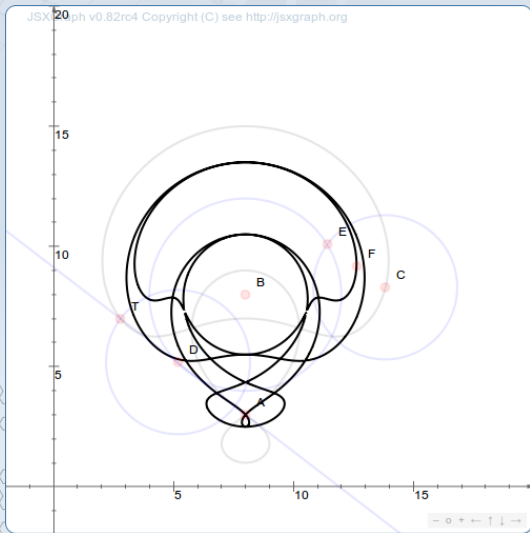
```
E = board.create('intersection', [c1, c2, 0]);
```

```
F = board.create('midpoint', [C, E]);
```

```
var brd = JXG.JSXGraph.initBoard('box', {ax:  
var s = brd.createElement('slider', [[1,3], [5  
var a = brd.createElement('slider', [[1,2], [5  
var b = brd.createElement('slider', [[1,1], [5  
var f = function(x) { return Math.sin(x); }  
var plot = brd.createElement('functiongraph',  
var os = brd.createElement('riemannsum', {f:  
function() { return s.Value(); }  
function() { return a.Value(); }  
function() { return b.Value(); }  
},  
{fillColor: '#ffff00';
```



# Implementation



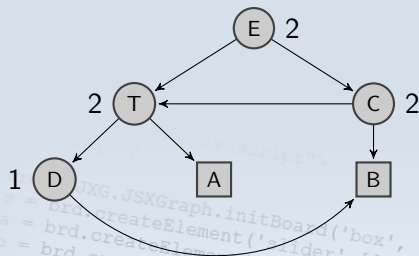
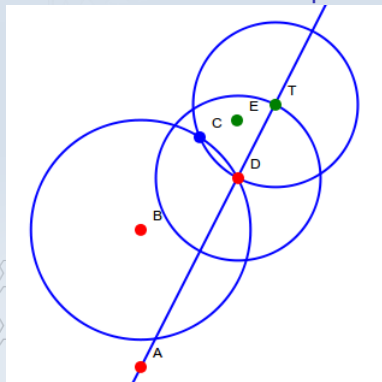
## Optimization

```
<script src="/javascript">  
var brd = JSXGraph.initBoard('box', {ax:  
var s = brd.createElement('slider', [[1,3],[5  
var a = brd.createElement('slider', [[1,2],[5  
var b = brd.createElement('slider', [[1,1],[5  
var f = function(x){ return Math.sin(x); }  
var plot = brd.createElement('functiongraph',  
var os = brd.createElement('riemannsum', {f:  
function(){ return s.Value(); }  
function(){return a.Value(); }  
function(){return b.Value(); }  
},  
{fillColor:'#ffff00'
```



# Optimization

Idea: Divide and conquer





# Optimization

## Transformations

- ▶ Translate the construction moving one point to  $(0,0)$
- ▶ Rotate the construction moving another point onto the x-axis
- ▶ After the Gröbner basis is calculated, the result is retransformed
- ▶ User can choose the two points or
- ▶ JSXGraph chooses two points (but sometimes not the best suited ones)



## Examples

```
<script src="/javascript">  
var brd = JXG.JSXGraph.initBoard('box', {ax:  
var s = brd.createElement('slider', [[1,3],[5  
var a = brd.createElement('slider', [[1,2],[5  
var b = brd.createElement('slider', [[1,1],[5,  
var f = function(x){ return Math.sin(x); }  
var plot = brd.createElement('functiongraph',  
var os = brd.createElement('riemannsum', {f:  
function(){ return s.Value(); }  
function(){return a.Value(); }  
function(){return b.Value(); }  
},  
{fillColor:'#ffff00'
```



Last slide

Thank You

- ▶ <http://jsxgraph.org/>
- ▶ <http://jsxgraph.uni-bayreuth.de/wiki/>

```
<script type="text/javascript">  
var brd = JXG.JSXGraph.initBoard('box', {ax:  
var s = brd.createElement('slider', [[1,3],[5  
var a = brd.createElement('slider', [[1,2],[5  
var b = brd.createElement('slider', [[1,1],[5  
var f = function(x){ return Math.sin(x); }  
var plot = brd.createElement('functiongraph',  
var os = brd.createElement('riemannsum', {f:  
function(){ return s.Value(); }  
function(){return a.Value(); }  
function(){return b.Value(); }  
},  
{fillColor:'#ffff00',
```



UNIVERSITÄT  
BAYREUTH