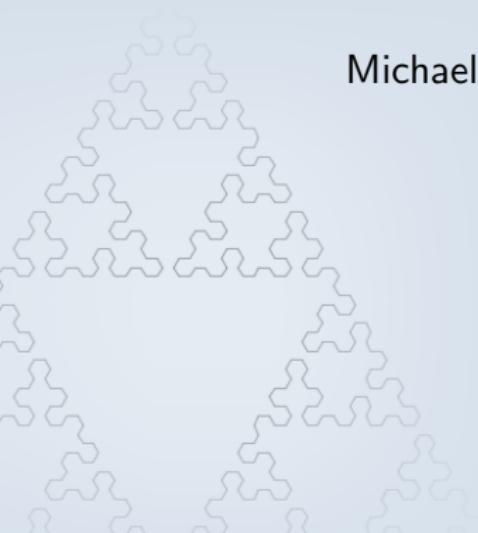




Automatic calculation of plane loci using Gröbner bases and integration into a Dynamic Geometry System



Michael Gerhäuser, Alfred Wassermann

July 24, 2010

```
brd = JXG.JSXGraph.initBoard('box', [ax, ay])
s = brd.createElement('slider',[[-1,3],[5,3],0])
a = brd.createElement('slider',[[-1,2],[5,2],0])
b = brd.createElement('slider',[[-1,1],[5,1],0])
plot = brd.createElement('functiongraph',
    function(x){ return Math.sin(x); })
r = brd.createElement('riemannsum', [f,
    s.Value(), a.Value(), b.Value(),
    10, true])
fillColor:'#ffff00',
```



UNIVERSITÄT
BAYREUTH

Overview

JSXGraph - A short overview

Computing plane loci using Gröbner bases

Optimizations

Implementing this algorithm in JSXGraph

```
<script>
  var board = JXG.JSXGraph.initBoard('box', [ax,
    bx, cx], {
      axis: true, grid: true, showLabels: false
    });
  var s = brd.createElement('slider',[[-1,3],[5,
    3],0]);
  var a = brd.createElement('slider',[[-1,2],[5,
    2],0]);
  var b = brd.createElement('slider',[[-1,1],[5,
    1],0]);
  var f = function(x){ return Math.sin(x); }
  var plot = brd.createElement('functiongraph',
    [f, a.Value(), b.Value()]);
  var riemannsum = brd.createElement('riemannsum',
    [f, s.Value(), a.Value(), b.Value(),
    10, '#ffff00']);
  var os = brd.createElement('riemannsum',
    [f, s.Value(), a.Value(), b.Value(),
    10, '#ffff00']);
  function() { return s.Value(); }
  function() { return a.Value(); }
  function() { return b.Value(); }
}, {fillColor:'#ffff00'});

</script>
```



JSXGraph



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



Supported Browsers

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



```
<script type="text/javascript">
  brd = JXG.JSXGraph.initBoard('box', {axis: true});
  s = brd.createElement('slider',[[-1,3],[5,3],0]);
  a = brd.createElement('slider',[[-1,2],[5,2],0]);
  b = brd.createElement('slider',[[-1,1],[5,1],0]);
  f = function(x){ return Math.sin(x); }
  plot = brd.createElement('functiongraph', [f]);
  r = brd.createElement('riemannsum', [f, a, b, 10]);
  r.os = brd.createElement('riemannsum', [f, a, b, 100]);
  r.function() = s.Value();
  r.function() = a.Value();
  r.function() = b.Value();
  r.fillStyle = '#ffff00';
</script>
```

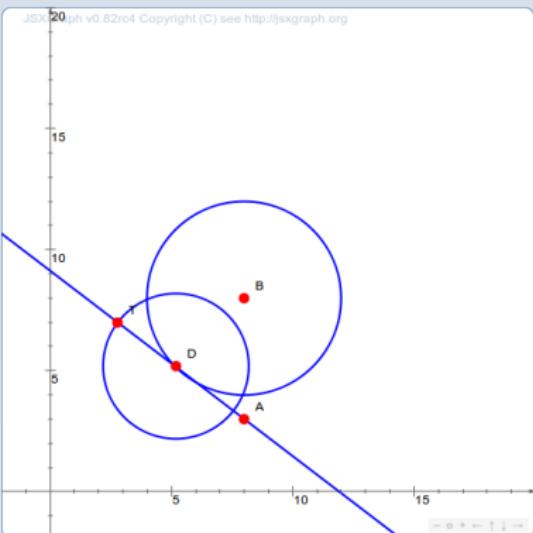


Example/Input

```
<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'});
</script>
```



Example/Output



```
JSXGraph v0.82rc4 Copyright (C) see http://jsxgraph.org
<script>
Graph.initBoard('box', [ax,
Element('slider',[[1,3],[5,
Element('slider',[[1,2],[5,
Element('slider',[[1,1],[5,
{ return Math.sin(x);
teElement('functiongraph',
prd.createElement('riemannsum',[f
function(){ return s.Value();
function(){ return a.Value();
function(){ return b.Value();
}, [
fillColor:'#ffff00,
```



Supported file formats

- ▶ GEONExT
- ▶ GeoGebra
- ▶ Intergeo
- ▶ Cinderella (small feature subset)

```
<script type="text/javascript">
  brd = JXG.JSXGraph.initBoard('box', [ax,
    [s = brd.createElement('slider',[[-1,3],[5,3],0.5]),
     a = brd.createElement('slider',[[-1,2],[5,2],1.5),
     b = brd.createElement('slider',[[-1,1],[5,1],2.5),
     f = function(x){ return Math.sin(x); },
     plot = brd.createElement('functiongraph',
       [f, s.Value()]),],
    [os = brd.createElement('riemannsum',[f,
      function(){ return s.Value(); },
      function(){ return a.Value(); },
      function(){ return b.Value(); },
      1,
      {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 = JXG.JSXGraph.loadBoardFromFile('jxgbox', 'watt.cdy', 'cinderella');

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

```
    <!-- slider for a -->
    var slider_a = brd.createElement('slider', [[1,3],[5,3],[3,3]], {name:'a'});
    var slider_b = brd.createElement('slider', [[1,2],[5,2],[3,2]], {name:'b'});

    if (f = function(x){ return Math.sin(x); })
        plot = brd.createElement('functiongraph',
            {function:f, xRange:[-1,1], yRange:[-1,1]}, {name:'f'});

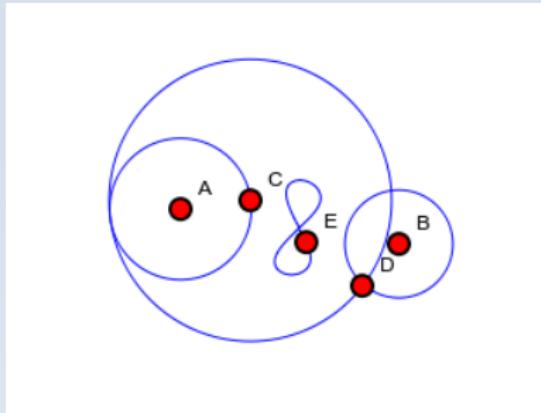
    os = brd.createElement('riemannsum', [f, a, b], {name:'os'});

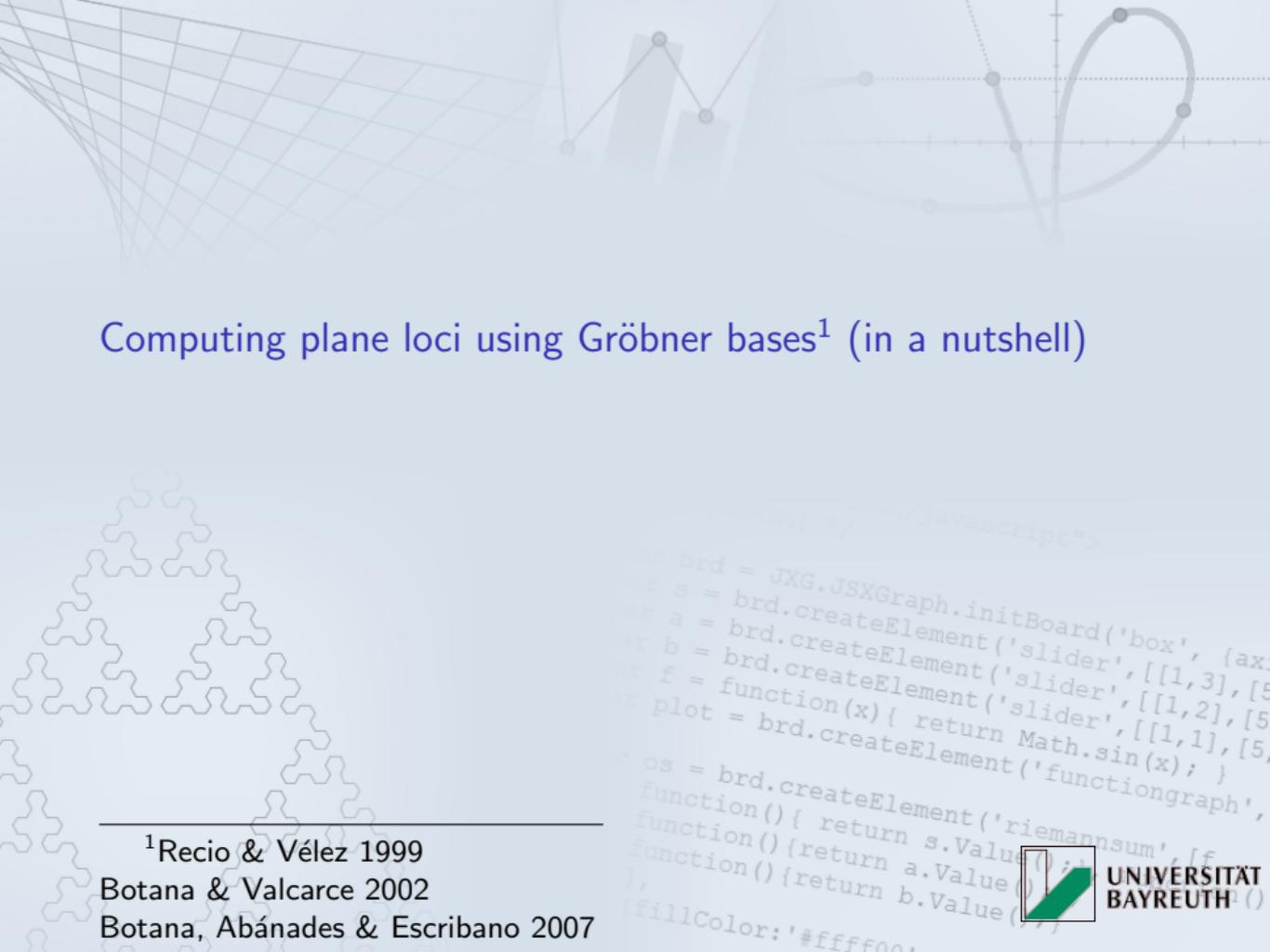
    function() { return s.Value(); }
    function() { return a.Value(); }
    function() { return b.Value(); }

    fillColor:'#ffff00',
```



Example/Output





Computing plane loci using Gröbner bases¹ (in a nutshell)

¹Recio & Vélez 1999

Botana & Valcarce 2002

Botana, Abánades & Escribano 2007

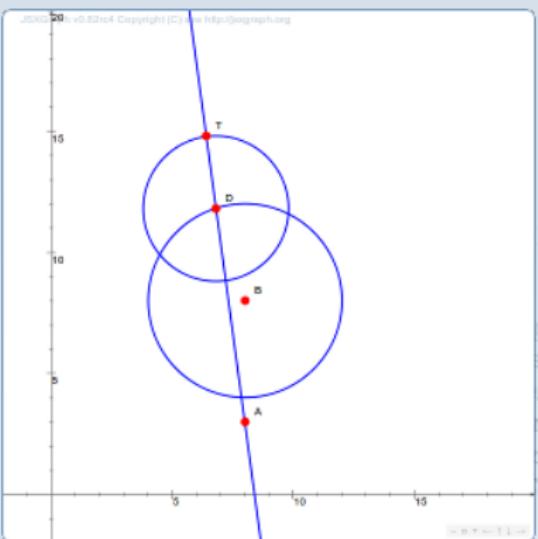
```
for brd = JXG.JSXGraph.initBoard('box', [ax, ay], {axis: true}), s = brd.createElement('slider', [[1, 3], [5, 3], [3, 3]]), a = brd.createElement('slider', [[1, 2], [5, 2], [3, 2]]), b = brd.createElement('slider', [[1, 1], [5, 1], [3, 1]]), f = function(x){ return Math.sin(x); }, plot = brd.createElement('functiongraph', [f]), riemannsum = brd.createElement('riemannsum', [f, s, a, b]), os = brd.createElement('riemannsum', [f, s, a, b]), os.Value = 0; s.on('change', function(){ os.Value = 0; riemannsum.update(); }), a.on('change', function(){ os.Value = 0; riemannsum.update(); }), b.on('change', function(){ os.Value = 0; riemannsum.update(); }), plot.on('click', function(){ os.Value = 0; riemannsum.update(); })},
```



UNIVERSITÄT
BAYREUTH

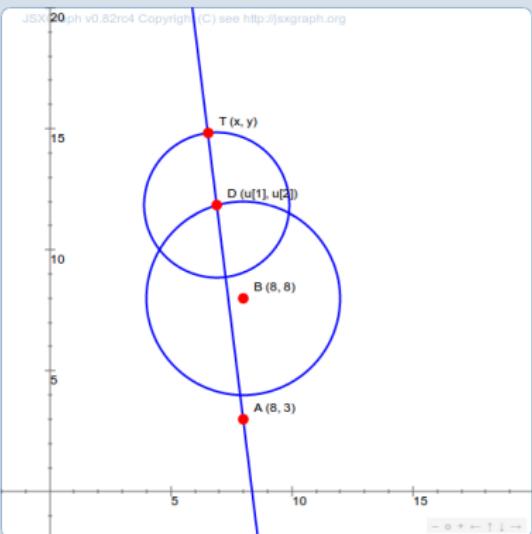
Computing plane loci using Gröbner bases

- ▶ Given a set of free and dependent points,



Computing plane loci using Gröbner bases

- ▶ we first choose a coordinate system,



```
JSXGraph v0.82rc4 Copyright (C) see http://jsxgraph.org
ph.initBoard('box', [ax,
element('slider',[[1,3],[5,
element('slider',[[1,2],[5,
element('slider',[[1,1],[5,
return Math.sin(x);
element('functiongraph',
function() { return s.Value();
function() { return a.Value();
function() { return b.Value();
}, [
fillColor:'#ffff00',
```



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 elimination ideal using the Gröbner basis of the given ideal,

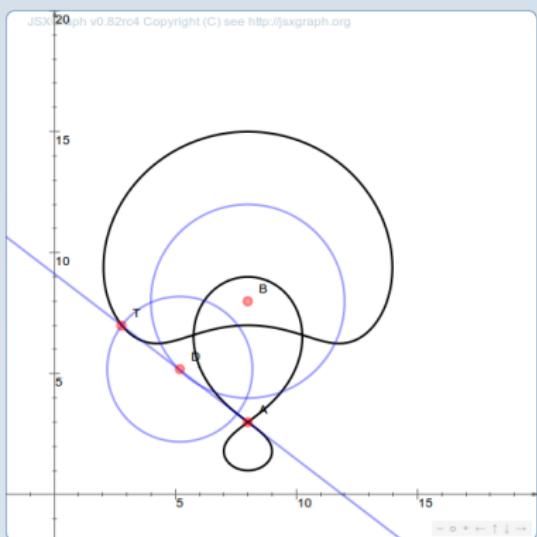
- ▶
$$x^6 + 3x^4y^2 + 3x^2y^4 + y^6 - 48x^5 - 38x^4y - 96x^3y^2 - 76x^2y^3 - 48xy^4 - 38y^5 + 1047x^4 + 1216x^3y + 1774x^2y^2 + 1216xy^3 + 727y^4 - 13024x^3 - 16596x^2y - 16096xy^2 - 8404y^3 + 97395x^2 + 109888xy + 63535y^2 - 415536x - 300806y + 790009 = 0$$

```
brd = board('box', [ax, ay], [bx, by], [cx, cy], [dx, dy])
a = brd.createElement('slider', [[1,3],[5,10]], [1, 5])
b = brd.createElement('slider', [[1,2],[5,10]], [1, 5])
f = function(x){ return Math.sin(x); }
plot = brd.createElement('functiongraph', [f], [0, 10], [0, 10])
os = brd.createElement('riemannsum', [f], [0, 10], [0, 10])
s = slider([0, 10], [0, 10], [0, 10])
a.Value = s.Value
b.Value = s.Value
function(){ return a.Value; }
function(){ return b.Value; }
},
fillColor: '#ffff00',
```



Computing plane loci using Gröbner bases

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



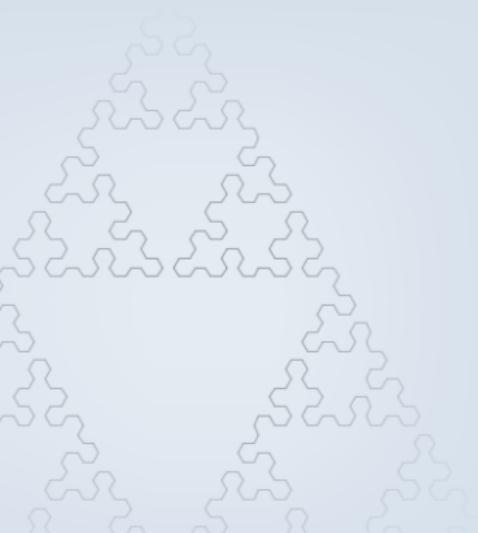
```
JSXGraph v0.82rc4 Copyright (C) see http://jsxgraph.org
ph.initBoard('box', [ax,
element('slider',[[1,3],[5,
element('slider',[[1,2],[5,
element('slider',[[1,1],[5,
return Math.sin(x);
element('functiongraph',
function() { return s.Value();
function() { return a.Value();
function() { return b.Value();
}, fillColor: '#ffff00,
```



UNIVERSITÄT
BAYREUTH



Implementing this algorithm in JSXGraph



```
<script type="text/javascript">
  var brd = JXG.JSXGraph.initBoard('box', [ax,
    s = brd.createElement('slider',[[1,3],[5,
    a = brd.createElement('slider',[[1,2],[5,
    b = brd.createElement('slider',[[1,1],[5,
    f = function(x){ return Math.sin(x);
    plot = brd.createElement('functiongraph',
    * 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 type="text/javascript">
    var brd = JXG.JSXGraph.initBoard('box', [ax,
        [s = brd.createElement('slider',[[-1,3],[5,3],0.5]),
        a = brd.createElement('slider',[[-1,2],[5,2],0.5]),
        b = brd.createElement('slider',[[-1,1],[5,1],0.5]),
        f = function(x){ return Math.sin(x); },
        plot = brd.createElement('functiongraph',
            [f, s, a, b])],
        [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

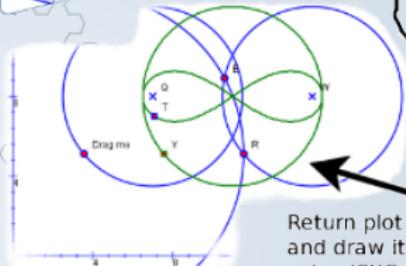
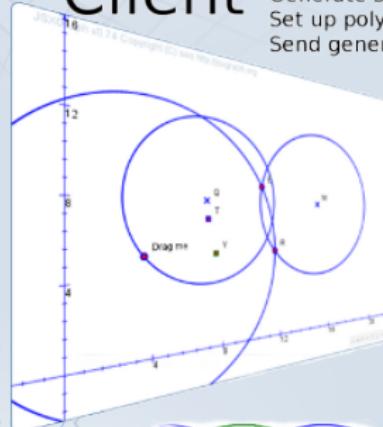
This enables us to

- ▶ 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



Return plot data
and draw it
using JSXGraph.

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.

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

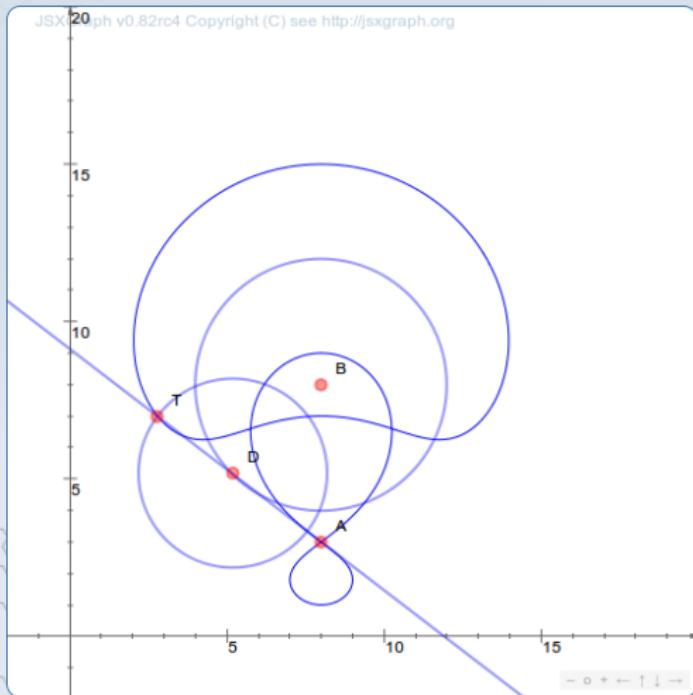
```
<link rel="stylesheet" type="text/css" href="css/jsxgraph.css" />
<script type="text/javascript" src="js/jsxgraphcore.js"></script>
...
<div id="jxgbox" class="jxgbox" style="width:500px; height:500px;"></div>
<script type="text/javascript">
  board = JXG.JSXGraph.initBoard('jxgbox', {boundingbox: [-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



UNIVERSITÄT
BAYREUTH

Implementation

Ready-to-use elements

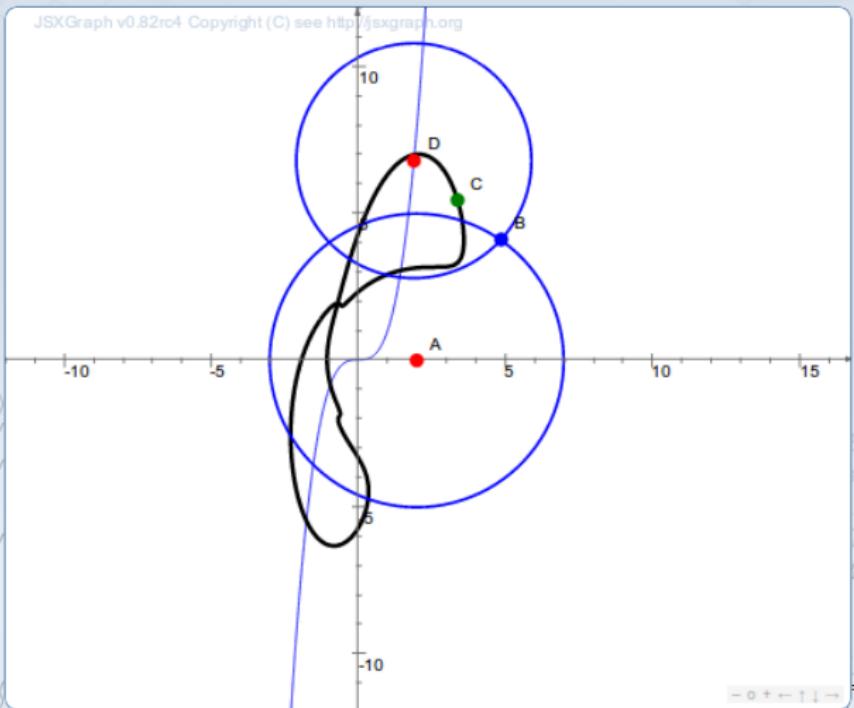
- ▶ Glider on circle and line
- ▶ Intersection points (circle/circle, circle/line, line/line)
- ▶ Midpoint
- ▶ Parallel line and point
- ▶ Perpendicular line and point
- ▶ Circumcircle and circumcenter

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



Implementation

Easy to extend



```
Board('box', [ax,  
lider',[[1,3],[5  
lider',[[1,2],[5  
lider',[[1,1],[5  
path.sin(x);  
'functiongraph',  
emarnsum',[f  
-o+-+11+  
"() {return b.Value();}  
fillColor:'#ffff00',
```



UNIVERSITÄT
BAYREUTH

Implementation

```
<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:[-4, 6, 8, -4], axis:
        true, grid: false, keepaspectratio: true});

    A = board.create('point', [2,0]);
    k1 = board.create('circle', [A, 5]);
    c = board.create('functiongraph', [function (x) { return x***; }]);

    c.generatePolynomial = function(p) {
        return [('+p.symbolic.x+')^3 - '+p.symbolic.y];
    };

    D = board.create('glider', [0,0, c], {name: 'D'});
    k2 = board.create('circle', [D, 4]);
    I = board.create('intersection', [k1, k2, 0]);

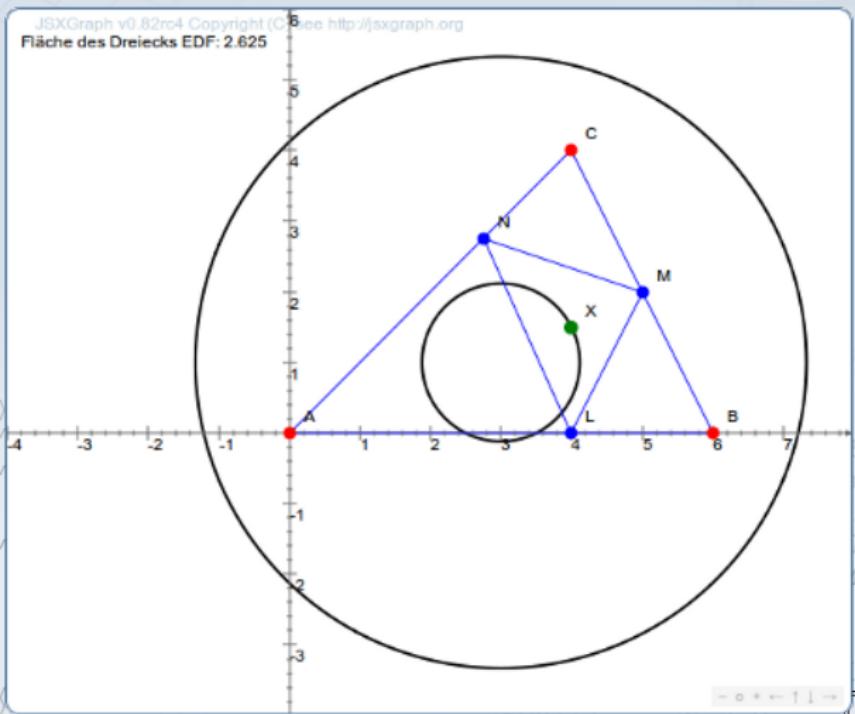
    C = board.create('midpoint', [I, D]);

    locus = board.create('locus', [C]);
</script>
```



Implementation

Easy to extend



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

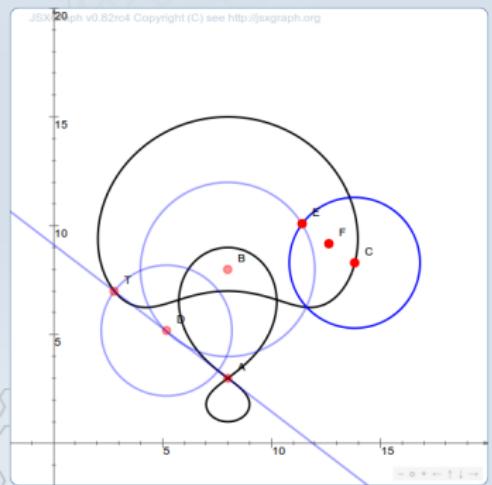
Re-using locus data: Discovered loci can be

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

```
s = JXG.JSXGraph.initBoard('box', [ax, ay])
var brd = s;
var a = brd.createElement('slider',[[-1,3],[5,3],0]);
var b = brd.createElement('slider',[[-1,2],[5,2],0]);
var f = function(x){ return Math.sin(x); }
var plot = brd.createElement('functiongraph',
    {f:f});
var riemannsum = brd.createElement('riemannsum',
    {f:f, a:a, b:b, n:100, stepColor:'#ffff00',
    fillColor:'#ffff00'});
var os = brd.createElement('riemannsum',
    {f:f, a:a, b:b, n:100, stepColor:'#0000ff',
    fillColor:'#0000ff'});
```



Implementation

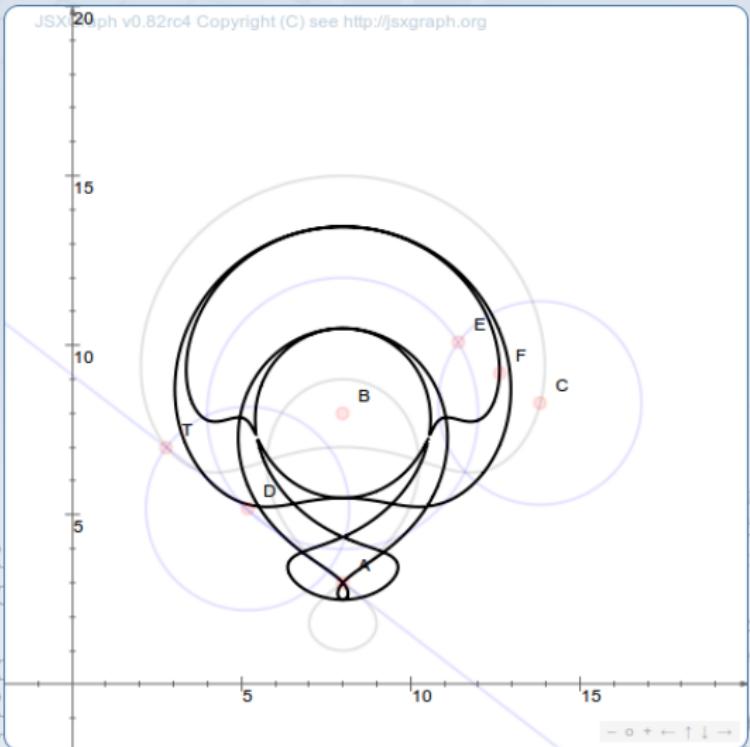


```
C = board.create('glider', [locus]);
c2 = board.create('circle', [C, 3]);
E = board.create('intersection', [c1, c2, 0]);
F = board.create('midpoint', [C, E]);
```

```
<script>
var brd = JXG.JSXGraph.initBoard('box', [ax,
  ay], {axis: true});
var s = brd.createElement('slider', [[1,3],[5,15], 10], {name: 's'});
var a = brd.createElement('slider', [[1,2],[5,15], 10], {name: 'a'});
var b = brd.createElement('slider', [[1,1],[5,15], 10], {name: 'b'});
var f = function(x){ return Math.sin(x); }
var plot = brd.createElement('functiongraph',
  [f, -pi, pi], {strokeWidth: 2});
var os = brd.createElement('riemannsum', [f, -pi, pi], {nRects: 10});
os.on('click', function() {
  var sValue = s.Value();
  var aValue = a.Value();
  var bValue = b.Value();
  if (sValue > aValue) {
    os.setRects(aValue, sValue);
  } else {
    os.setRects(sValue, bValue);
  }
}, {fillColor: '#ffff00', strokeDash: [5, 5]});</script>
```



Implementation



UNIVERSITÄT
BAYREUTH

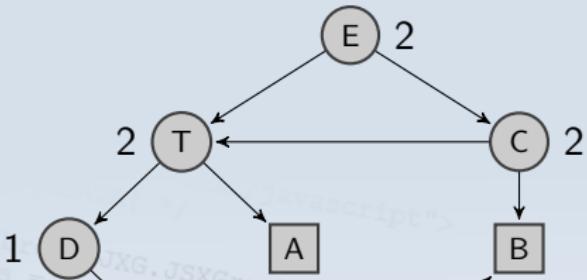
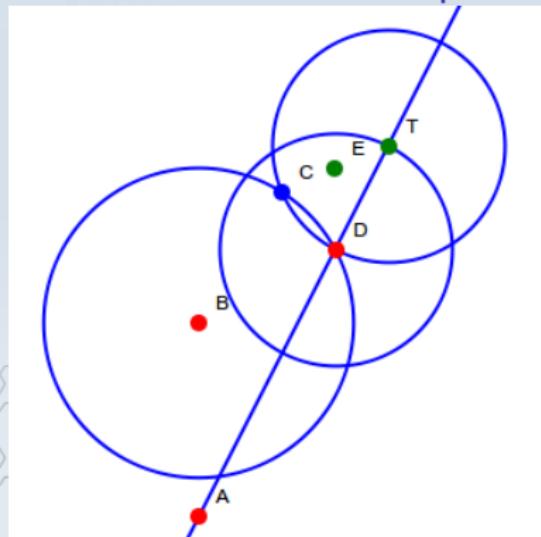
Optimization

```
<script type="text/javascript">
var brd = JXG.JSXGraph.initBoard('box', {axis: true});
var s = brd.createElement('slider',[[-1,3],[5,-1],0]);
var a = brd.createElement('slider',[[-1,2],[5,-1],0]);
var b = brd.createElement('slider',[[-1,1],[5,-1],0]);
var f = function(x){ return Math.sin(x); }
var plot = brd.createElement('functiongraph', [f]);
var os = brd.createElement('riemannsum', [f]);
os.on('click', function() { return s.Value(); });
os.on('start', function() {return a.Value(); });
os.on('end', function() {return b.Value(); });
}, {fillColor:'#ffff00'});

</script>
```

Optimization

Idea: Divide and conquer



```
g = brd.createBoard('box', [ax, ay])
a = brd.createElement('slider',[[-1,3],[5,10]], 'a')
b = brd.createElement('slider',[[-1,2],[5,10]], 'b')
f = function(x){ return Math.sin(x); }
plot = brd.createElement('functiongraph',
    {os: brd.createElement('riemannsum', [f,
        function(){ return s.Value(); },
        function(){ return a.Value(); },
        function(){ return b.Value(); },
        {fillColor:'#ffff00'}])});
```



Optimization

Transformations

- ▶ Translate the construction moving one point to $(0, 0)$
- ▶ Rotate the construction around the origin, 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)



Last slide

Thank You

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

```
var brd = JXG.JSXGraph.initBoard('box', [ax, ay]);
var s = brd.createElement('slider',[[-1,3],[5,3],0]);
var a = brd.createElement('slider',[[-1,2],[5,2],0]);
var b = brd.createElement('slider',[[-1,1],[5,1],0]);
var f = function(x){ return Math.sin(x); }
var plot = brd.createElement('functiongraph',
    {f:f, x1:a.Value(), x2:b.Value()});
var riemannsum = brd.createElement('riemannsum',
    {x1:a.Value(), x2:b.Value(), n:100});
var os = brd.createElement('riemannsum',
    {x1:a.Value(), x2:b.Value(), n:100,
        function(){ return s.Value(); },
        function(){ return a.Value(); },
        function(){ return b.Value(); }});
os.FillColor = '#ffff00';
```



UNIVERSITÄT
BAYREUTH