Welcome to MCS 563

1. About the Course
   - content and organization
   - expectations of the course

2. Solving Polynomial Systems
   - representation of polynomials
   - an application: inverse kinematics
   - a local approach: Newton's method
   - statement of Bézout's theorem

MCS 563 Lecture 1
Analytic Symbolic Computation
Jan Verschelde, 10 January 2011

Welcome to MCS 563

Catalog Description

outdated

MCS 563 - Analytic Symbolic Computation

Surviving course, as Algebraic Symbolic Computation, formerly known as MCS 561 has vanished.

Prerequisite(s): the following courses are relevant: MCS 320 (Introduction to Symbolic Computation), MCS 471 (Numerical Analysis), MATH 330 (Abstract Algebra).

A better title for the course could be:

symbolic-numeric methods in algebraic geometry.

The course is part of the computational science prelim.

Content of the Course

no Text Book

A first run of the course (Fall 2004) followed Hans J. Stetter: *Numerical Polynomial Algebra*, SIAM 2004.

Lecture notes were developed in Spring 2007 and 2009.

Computational algebraic geometry is well developed but relied (in the past) mostly on symbolic methods.

Guiding problem: solve polynomial systems.

Interlacing symbolic and numeric methods:

1. at first separately, alternatingly,
2. integrating algorithms into a solving methodology.

Welcome to MCS 563

1 About the Course
content and organization
expectations of the course

2 Solving Polynomial Systems
representation of polynomials
an application: inverse kinematics
a local approach: Newton's method
statement of Bézout's theorem

Organization and Expectations

The first half of the course: basic methods and concepts, followed by review and midterm (or take home exam, extended homework assignment).

First project involves exploratory use of computer algebra. Second project could be literature study in preparation of the final project, along with more specialized topics in the second part of the course.

Expectations of the course:

- prepare for the computational science prelim,
- introduction to research topics.

Homework is important if interested in prelim;
instead of final exam, we may opt for research projects.

Welcome to MCS 563

polynomial systems

Given is a system $f(\mathbf{x}) = \mathbf{0}$ of N polynomials
$f = (f_1, f_2, \ldots, f_N)$ in n unknowns $\mathbf{x} = (x_1, x_2, \ldots, x_n)$.
The coefficients are in $\mathbb{C}$, exact or approximate.

Three cases:

1. $N > n$: f is overdetermined, or
2. $N < n$: f is underdetermined, or
3. $N = n$: f is square.

The polynomials in f can be

- dense: degree is main characteristic,
- sparse: the support A is a set of exponent vectors,
- functions to evaluate the polynomials.

representation of polynomials

- Dense polynomials are often of low degree, because #monomials grows exponentially; as Pascal's formula:

$$\binom{d+n}{d} = \binom{d+(n-1)}{d} + \binom{(d-1)+n}{d-1}.$$

- Let A be a finite subset of $\mathbb{N}^n$, in multi-index notation:

$$p(\mathbf{x}) = \sum_{\mathbf{a}\in A} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}}, \quad c_{\mathbf{a}} \in \mathbb{C}, \quad \mathbf{x}^{\mathbf{a}} = x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}.$$

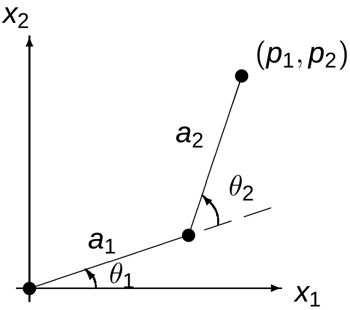
  E.g.: $p(x_1, x_2) = c_{12}x_1x_2^2 + c_{20}x_1^2 + c_{11}x_1x_2 + c_{00}$.
  The convex hull of A is the Newton polytope.

- Finding the best way to evaluate polynomials is hard.

Welcome to MCS 563

Inverse Kinematics

Consider a planar robot arm, with two links and two joints:



The lengths a_1 and a_2 of the links are parameters.
Variables are the angles θ_1 and θ_2 at the joints.
The position of the hand are in the coordinates (p_1, p_2).

Forward kinematics: given (θ_1, θ_2), determine (p_1, p_2).
Inverse kinematics: given (p_1, p_2), find (θ_1, θ_2).

changing coordinates

Consider coordinate transformations so new origin is first joint, rotated so first link is first coordinate axis:

$$\begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\textit{rotate by } \theta_1} \underbrace{\begin{bmatrix} 1 & 0 & a_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\textit{translate}} \begin{bmatrix} x_1' \\ x_2' \\ 1 \end{bmatrix} \quad \begin{array}{l} s_1 = \sin(\theta_1), \\ c_1 = \cos(\theta_1). \end{array}$$

Denote product of the rotation-translation matrices by

$$A_1 = \begin{bmatrix} c_1 & -s_1 & c_1 a_1 \\ s_1 & c_1 & s_1 a_1 \\ 0 & 0 & 1 \end{bmatrix}$$

and similarly denote A_2 for the coordinate transformation using the second link and the second angle.

polynomial equations

Using A_1 and A_2:

$$\begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = A_1 A_2 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} c_1 c_2 a_2 - s_1 s_2 a_2 + c_1 a_1 \\ s_1 c_2 a_2 + c_1 s_2 a_2 + s_1 a_1 \\ 1 \end{bmatrix},$$

expresses that the hand of the robot has coordinates $(0, 0)$ in the third coordinate system and (x_1, x_2) in the first coordinate system. So the polynomial system is

$$\left\{ \begin{array}{r} c_1 c_2 a_2 - s_1 s_2 a_2 + c_1 a_1 = p_1 \\ s_1 c_2 a_2 + c_1 s_2 a_2 + s_1 a_1 = p_2 \\ c_1^2 + s_1^2 = 1 \\ c_2^2 + s_2^2 = 1 \end{array} \right.$$

where the last two equations are $\sin^2(\theta_1) + \cos^2(\theta_1) = 1$ and $\sin^2(\theta_2) + \cos^2(\theta_2) = 1$.

Welcome to MCS 563

Newton's method

Often we have already a good guess for a solution.
For $n = 1$, the Taylor expansion of a function f about x is

$$f(x+\Delta x) = f(x) + \Delta x \frac{f'(x)}{1!} + (\Delta x)^2 \frac{f''(x)}{2!} + (\Delta x)^3 \frac{f'''(x)}{3!} + \cdots$$

If we truncate the expansion at the second term:

$$f(x + \Delta x) = f(x) + \Delta x f'(x) + O((\Delta x)^2).$$

Solve for Δx in $f(x + \Delta x) = 0$: $\Delta x = -f(x)/f'(x)$.

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)}, \quad \text{for } k = 0, 1, \ldots$$

is well defined provided $f'(x_k) \neq 0$ for all k.

a script to run Newton's method

```
function [x,fail] = newton(f,df,x,eps,N)
%
%  Applies Newton's method to the function f,
%  with derivative in df, starting at the point x.
%  It produces a sequence of approximations.
%  Stops at x where |f(x)| < eps or when two
%  consecutive points in the sequence are less
%  than eps apart from each other.
%  On return, fail = 1 if the accuracy requirement
%  is not satisfied in N steps; otherwise fail = 0.
%
%  Example :
%  >> [x,fail] = newton('sin','cos',pi/4,1e-8,10)
%
```

For builtin functions we can use strings,
otherwise we define `f` and `df` via `inline` or m-files.

code for the script

```
function [x,fail] = newton(f,df,x,eps,N)
%
fprintf('running the method of Newton...\n');
fx = feval(f,x);
for i = 1:N
   dx = fx/feval(df,x);
   x = x - dx;
   fx = feval(f,x);
   fprintf('x = %.4e, dx = %.3e, f(x) = %.2e\n',
           x, dx, fx);
   if (abs(fx) < eps) | (abs(dx) < eps)
      fail = 0;
      fprintf('succeeded after %d steps\n', i);
      return;
   end
end
fprintf('failed requirements after %d steps\n', N);
fail = 1;
```

running Newton's method

Octave is an open source alternative to MATLAB.

```
octave-3.2.3:1>
[x,fail] = newton('sin','cos',pi/4,1.0e-8,10)
running the method of Newton...
x = -2.1460e-01, dx = 1.000e+00, f(x) = -2.13e-01
x = 3.3563e-03, dx = -2.180e-01, f(x) = 3.36e-03
x = -1.2602e-08, dx = 3.356e-03, f(x) = -1.26e-08
x = 1.6544e-24, dx = -1.260e-08, f(x) = 1.65e-24
succeeded after 4 steps
x =  1.6544e-24
fail = 0
```

Observe the quadratric convergence:

$$\text{if } f(x^*) = 0\text{: } ||x_{k+1} - x^*|| \leq ||x_k - x^*||^2.$$

Newton's method in several variables

The Taylor expansions in matrix form:

$$f(\mathbf{x}+\Delta\mathbf{x}) = f(\mathbf{x}) + \underbrace{\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_n} \end{bmatrix}}_{Jacobian\ matrix\ J_f} (\Delta\mathbf{x}) + O((\Delta\mathbf{x})^2).$$

Solve for $\Delta\mathbf{x}$ in $f(\mathbf{x}+\Delta\mathbf{x}) = \mathbf{0}$: $J_f(\mathbf{x})\Delta\mathbf{x} = -f(\mathbf{x})$.

Newton's method produces a sequence of approximations:

$$\begin{cases} J_f(\mathbf{x}_k)\Delta\mathbf{x} = -f(\mathbf{x}_k) \\ \mathbf{x}_{k+1} := \mathbf{x}_k + \Delta\mathbf{x} \end{cases} \quad k = 0, 1, \ldots$$

what is a solution?

A solution $\mathbf{x}^*$ of a square system $f(\mathbf{x}) = \mathbf{0}$ is

- regular if $\det(J_f(\mathbf{x}^*)) \neq 0$,
- singular if $\det(J_f(\mathbf{x}^*)) = 0$.

Multiple and nonisolated solutions are singular.

A solution $\mathbf{x}^*$ is isolated if there exists a ball centered at $\mathbf{x}^*$ with nonzero radius that contains no other solution.

Real solutions may occur as isolated real points on a complex curve, e.g.: $(0, 0)$ on $x^2 + y^2 = 0$.

Welcome to MCS 563

Bézout's theorem

How many solutions?

Theorem (the fundamental theorem of algebra)

A polynomial of degree d in one variable with nonzero leading coefficient has exactly d complex roots, where each root is counted with its multiplicity.

e.g.: $p(x) = x(x-1)^3$, we count 1 three times.

Theorem (theorem of Bézout on #isolated solutions)

Let $f(\mathbf{x}) = \mathbf{0}$ be a square system of dimension n. The number of isolated solutions in $\mathbb{C}^n$ is bounded by

$$D = \prod_{i=1}^{n} \deg(f_i).$$

We call D the total degree of the system.

Summary + Exercises

We defined the input to our problems, gave an application, recalled Newton's method and stated Bézout's theorem.

Exercises:

1. The notion of a sparse polynomial changes when one considers the number of vertices which span its Newton polygon, i.e.: a dense polynomial may have a Newton polygon which is spanned by only a few vertices. Give an example of two polynomials which have the same Newton polygon where one polynomial is sparse and the other is dense.
2. How many arithmetical operations are needed to evaluate a polynomial of degree d in n variables with a Horner scheme? Start with $n = 1$ and generalize.
3. How many solutions do you expect to the inverse position problem for the planar robot arm with two links? Make a case analysis.

more exercises

4. Use Maple to solve the inverse position problem for the planar robot arm with two links? In your answer, give the Maple instructions you used and report on the results Maple returns to you. Open source alternatives to Maple, e.g.: CoCoA, Macaulay 2, Sage, or Singular may be used as well.
5. Generalize the MATLAB script `newton` so that it also works for systems. You can of course also use Octave. Give a numerical example to illustrate that your generalized `newton` script works.
6. Apply Bézout's theorem to bound the number of isolated solutions to the system for the inverse position problem for the planar robot arm with two links. Can you derive an equivalent formulation for this problem which leads to a system with lower total degree? (Hint: consider multiplying the matrix equation by A_1^{-1}.)

and more exercises

7. Use Maple or Sage (free to download from `www.sagemath.org`) and make a worksheet or notebook to derive the polynomial equations for the inverse kinematics problem. Also derive the alternative formulation suggested in the previous exercise, multiplying with A^{-1}.

A selection of these exercises will be collected at a date to be announced soon.