Calcul parallèle exacte des solutions réelles de systèmes algébriques

Marc Moreno Maza

University of Western Ontario (Canada)

LIFL
Université Lille 1
19 Mai 2011

Plan

1. Computing the real solutions of polynomial systems symbolically
2. Hilbert 16's Problem
3. Real Root Isolation on Multicores
4. Solving Polynomial Systems on the GPU

Plan

1. Computing the real solutions of polynomial systems symbolically
2. Hilbert 16's Problem
3. Real Root Isolation on Multicores
4. Solving Polynomial Systems on the GPU

A biochemical network: setting (1/3)

The generic kinetic scheme of prion diseases is illustrated as follows:

$$\begin{array}{c} \downarrow 1 \\ PrP^C \quad \xrightarrow{3} PrP^{Sc} \xrightarrow{4} \text{Aggregates.} \\ \downarrow 2 \end{array}$$

where

- $[PrP^C]$ is the concentration of PrP^C (harmless form)
- $[PrP^{Sc}]$ is the concentration of PrP^{Sc} (infectious form) which catalyses the transformation from the normal form to itself,
- Step 1: synthesis of native PrP^C
- Step 2, 4: normal degradation.

A biochemical network: setting (2/3)

Let ν_i be the rate of Step i for $i = 1, \ldots, 4$.

$$\begin{tikzcd}
{} \arrow[d, "1"] & & \\
PrP^C \arrow[r, "3"] \arrow[d, "2"] & PrP^{Sc} \arrow[r, "4"] & \text{Aggregates.} \\
{} & &
\end{tikzcd}$$

- Step 1: zero-order kinetic process, that is $\nu_1 = k_1$,
- Step 2, 4: first-order rate equations: $\nu_2 = k_2 \left[PrP^C\right]$, $\nu_4 = k_4 \left[PrP^{Sc}\right]$.
- Step 3: a nonlinear process

$$\nu_3 = \left[PrP^C\right] \frac{a\left(1 + b\left[PrP^{Sc}\right]^n\right)}{1 + c\left[PrP^{Sc}\right]^n}.$$

A biochemical network: setting (3/3)

$$\begin{tikzcd} \arrow[d, "1"] & & \\ PrP^C \arrow[r, "3"] \arrow[d, "2"] & PrP^{Sc} \arrow[r, "4"] & \text{Aggregates.} \\ {} & & \end{tikzcd}$$

We also have:

$$\frac{\mathrm{d}\left[PrP^C\right]}{\mathrm{d}t} = \nu_1 - \nu_2 - \nu_3$$
$$\frac{\mathrm{d}\left[PrP^{Sc}\right]}{\mathrm{d}t} = \nu_3 - \nu_4$$

Letting $x = \left[PrP^C\right]$ and $y = \left[PrP^{Sc}\right]$. we obtain the dynamical system:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = k_1 - k_2 x - ax\frac{(1+by^n)}{1+cy^n}$$
$$\frac{\mathrm{d}y}{\mathrm{d}t} = ax\frac{(1+by^n)}{1+cy^n} - k_4 y$$

A biochemical network: semi-algebraic systems to solve

Dynamical system to study

M. Laurent (Biochem. J., 1996) suggests to set $b = 2$, $c = 1/20$, $n = 4$, $a = 1/10$, $k_4 = 50$ and $k_1 = 800$, leading to:

$$\left\{ \begin{array}{ccc} \frac{\mathrm{d}x}{\mathrm{d}t} & = & f_1 \\ \frac{\mathrm{d}y}{\mathrm{d}t} & = & f_2 \end{array} \right. \quad \text{with} \quad \left\{ \begin{array}{ccc} f_1 & = & \frac{16000+800y^4-20k_2x-k_2xy^4-2x-4xy^4}{20+y^4} \\ f_2 & = & \frac{2(x+2xy^4-500y-25y^5)}{20+y^4} \end{array} \right. .$$

Semi-algebraic systems to solve

By Routh-Hurwitz criterion, an equilibrium (x, y) is asymptotically stable if

$$\Delta_1 := -\left(\frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y}\right) > 0 \text{ and } a_2 := \frac{\partial f_1}{\partial x} \cdot \frac{\partial f_2}{\partial y} - \frac{\partial f_1}{\partial y} \cdot \frac{\partial f_2}{\partial x} > 0.$$

Letting p_1, p_2 the above polynomials, we obtain two semi-algebraic systems:

$$\mathcal{S}_1 : \{p_1 = p_2 = 0, k_2 > 0\} \text{ and } \mathcal{S}_2 : \{p_1 = p_2 = 0, k_2 > 0, \Delta_1 > 0, a_2 > 0\}$$

A biochemical network: solving $\mathcal{S}_1$

The real solutions of $\mathcal{S}_1$ are described by the following **triangular decomposition** into **regular semi-algebraic systems**.

$$A_1 := \left\{ \begin{array}{ccc} (2y^4+1)x - 25y^5 - 500y & = & 0 \\ (k_2+4)y^5 - 64y^4 + (2+20k_2)y - 32 & = & 0 \\ k_2 & > & 0 \\ r & \neq & 0 \end{array} \right. , \quad A_2 := \left\{ \begin{array}{ccc} t_x & = & 0 \\ t_y & = & 0 \\ r & = & 0 \\ k_2 & > & 0 \end{array} \right. .$$

where $t_y(k_2, y)$ has degree 4 in y and r is given by

$$\begin{array}{rcl} r & := & 100000k_2^8 + 1250000k_2^7 + 5410000k_2^6 + 8921000k_2^5 - 9161219950k_2^4 \\ & - & 5038824999k_2^3 - 1665203348k_2^2 - 882897744k_2 + 1099528405056. \end{array}$$

The polynomial r has four real roots, two are positive: $\alpha_1 < \alpha_2$.

A biochemical network: solving $\mathcal{S}_1$

Regarding k_2 as a parameter, one can compute a **real comprehensive triangular decomposition** which gives:

$$\begin{cases} \{\ \} & k_2 \leq 0 \\ \{A_1\} & 0 < k_2 < \alpha_1 \\ \{A_2\} & k_2 = \alpha_1 \\ \{A_1\} & \alpha_1 < k_2 < \alpha_2 \\ \{A_2\} & k_2 = \alpha_2 \\ \{A_1\} & k_2 > \alpha_2 \end{cases} .$$

From where we deduce the number of real solutions:

$$\begin{cases} 0 & k_2 \leq 0 \\ 1 & k_2 > 0 \text{ and } r > 0 \\ 2 & k_2 > 0 \text{ and } r = 0 \\ 3 & k_2 > 0 \text{ and } r < 0 \end{cases}$$

A biochemical network: conclusion

Theorem

Assume that $k_2 > 0$. Then we have: if $r > 0$, then the dynamical system has 1 equilibrium; if $r = 0$, then it has 2 equilibria; if $r < 0$, it has 3 equilibria.

Theorem

Assume that $k_2 > 0$. Then we have: if $r > 0$, then the system has one *hyperbolic equilibrium, which is asymptotically stable; if $r < 0$ and $r_2 \neq 0$, then the system has* three *hyperbolic equilibria, two of which are asymptotically stable and the other one is unstable; if $r = 0$ or $r_2 = 0$, the system experiences a* bifurcation *where*

$$\begin{aligned} r_2 = {} & 10004737927168 k_2^9 + 624166300700672 k_2^8 + 7000539052537600 k_2^7 \\ & + 45135589467012800 k_2^6 - 840351411856453750 k_2^5 - 5009800435224844687 5k_2^4 \\ & - 27388168989455000000 k_2^3 - 86752092666960000000 k_2^2 \\ & + 102960917356800000000 k_2 + 59325460641024000000000. \end{aligned}$$

Summary and notes

- Solving for the **real roots** of (parametric or not) polynomial systems is a fundamental problem with many applications.
- Most of the time, this requires symbolic (and exact) computation.
- Computer algebra systems have limited capabilities for that, especially for the parametric case.
- Recent work (Changbo Chen, James H. Davenport, M^3 , Bican Xia & Rong Xiao ISSAC 2010-2011) is changing that.
- **RealTriangularize** is available MAPLE 15, as part of the MAPLE **RegularChains** library.

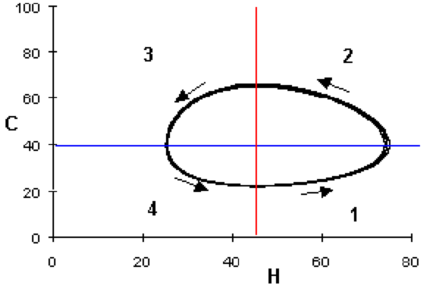
Plan

Cycles limite dans le modèle proie-prédateur

- Two species interact, one is a predator and one is its prey, according to the pair of differential equations:

$$\begin{cases} \frac{dx}{dt} = x(a - by) \\ \frac{dy}{dt} = -y(c - dx). \end{cases}$$

- Say x and y are numbers of **carnivores** and **herbivores**, while a, b, c, d are parameters.



At $(x, y) = (\frac{c}{d}, \frac{b}{a})$, we have a limit cycle:

- as the number of herbivores increases, then so does that of carnivores.
- but as that of carnivores increases, that of herbivores decreases, . . .

The statement

H 16: modern version

The (second half) of the 16th problem is one of the two remaining ones.

It asks for an upper bound of the number of limit cycles in polynomial vector fields:

$$\dot{x} = P_n(x, y), \quad \dot{y} = Q_n(x, y) \tag{1}$$

where $P_n(x, y)$ and $Q_n(x, y)$ are real polynomials of total degree n.

So far one got there:

$n = 2$ is solved and the maximum is 3.

But $n = 3$ resists, even if restricting to the nearby of isolated fixed points.

We consider the computation of small limit cycles bifurcated from a center at origin.

Problem set up

Original problem:

Consider a general normalized cubic system:

$$\begin{aligned}\dot{x} &= a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 + a_{30}x^3 + a_{21}x^2y + a_{12}xy^2 + a_{03}y^3 \\ \dot{y} &= b_{10}x + b_{01}y + b_{20}x^2 + b_{11}xy + b_{02}y^2 + b_{30}x^3 + b_{21}x^2y + b_{12}xy^2 + b_{03}y^3\,. \end{aligned} \tag{2}$$

Reworked problem:

After various transformations (rescaling, etc.) aiming at reducing the number of parameters, one obtains:

$$\begin{aligned}\dot{x} =&\alpha x + y + x^2 + (b+2d)xy + cy^2 + fx^3 + gx^2y + (h-3p)xy^2 + ky^3 \\ \dot{y} =& -x + \alpha y + dx^2 + (e-2)xy - dy^2 + lx^3 + (m-h-3f)x^2y \\ &\quad + (n-g)xy^2 + py^3. \end{aligned} \tag{3}$$

which depends on **13 variables** $\{\alpha, b, c, d, e, f, g, h, k, l, m, n, p\}$.

Using polar coordinates

Normal form

One obtains the so-called normal form:

$$\begin{aligned}\frac{dr}{dt} &= r(v_0 + v_1 r^2 + v_2 r^4 + \cdots + v_k r^{2k}),\\ \frac{d\theta}{dt} &= 1 + \omega + t_1 r^2 + t_2 r^4 + \cdots + t_k r^{2k},\end{aligned} \tag{4}$$

where $v_0, \ldots, v_k$ depend polynomially on $\{\alpha, b, c, d, e, f, g, h, k, l, m, n, p\}$.

Theorem (Yu Pei)

If the system

$$v_0 = v_1 = \cdots = v_{k-1} = 0,\ v_k \neq 0\,, \tag{5}$$

is consistent, then there are at most k limit cycles. Furthermore, these are **exactly** *k limit cycles if at one* **real** *solution we have:*

$$det\left(\frac{\partial v_i}{\partial a_j}\right)_{(k-1)\times(k-1)} \neq 0 \tag{6}$$

The system to solve

13 should be the maximum!

It follows that "generically"' we need to solve

$$v_0 = v_1 = \cdots = v_{k-1} = 0, \; v_k \neq 0 \,, \tag{7}$$

for $k = 13$, since there are 13 variables $\{\alpha, b, c, d, e, f, g, h, k, l, m, n, p\}$.

Thus we expect to prove that 13 is an upper bound.

The system is hard to generate

So far we could only generate $v_0, v_1, \ldots, v_9$ after several days of computation with MAPLE.

However, $v_0, v_1, \ldots, v_9$ appear to be **very sparse** (linear growth w.r.t. their total degree).

A first attempt via symbolic solving

Make the origin a center!

We return to the Cartesian formulation

$$\begin{aligned}\dot{x} &= ax + y + x^2 + (b+2d)xy + cy^2 + fx^3 + gx^2y + (h-3p)xy^2 + ky^3 \,, \\ \dot{y} &= -x + ay + dx^2 + (e-2)xy - dy^2 + lx^3 + (m-h-3f)x^2y + (n-g)xy^2 + py\end{aligned} \tag{8}$$

and set $\alpha = b = e = h = m = n = 0$, $p = f$ and
$n = 1/3(35c^2 + 30c - 15l - 15k - 45)$.

Experimental result

We solved the new system for $g < f < l < k < c$ modulo a 2^{58}-bit prime.
After 19 days of MAPLE, using 9506.1MB, we obtained **852 complex roots** using the `RegularChains library`.
Unfortunately, the output length is $6,355,573$ character long.
Too big for isolating the real roots on a desktop!

Summary and notes

- There is hope to solve Hilbert 16's Problem, for $n = 3$, on a cluster (but not on a desktop).
- Using symbolic computation is required.
- We are currently building a cluster and software for that purpose.
- Joint work (dynamical system part) with Changbo Chen, Robert M. Corless, Pei Yu, Yiming Zhang.
- The involved `RegularChains library` algorithms are based on the following papers:
    - (Changbo Chen & M^3 , ISSAC 2011)
    - (Xavier Dahan, M^3 , Éric Schost, Yuzhen Xie, ISSAC 2005)
    - (François Boulier, Changbo Chen, François Lemaire & M^3 , ASCM 2009)
    - (Changbo Chen, James H. Davenport, M^3 , Bican Xia & Rong Xiao, ISSAC 2010 & ISSAC 2011)

Plan

Reduction to Taylor shift

The Taylor shift $\mathbf{x} \longmapsto \mathbf{f}(\mathbf{x}+\mathbf{1})$ operation is at the core of Collins-Akritas Algorithm for real root isolation (counting).

Algorithm 1: NumberInZeroOne(p)

Input: a squarefree univariate polynomial p

Output: number of real roots of p in $(0, 1)$

```
1  begin
2      $p_1 := x^n p(1/x)$; $p_2 := \mathbf{p_1}(\mathbf{x}+\mathbf{1})$
3      let $d$ be the number of sign variations of the coefficients of $p_2$
4      if $d \leq 1$ then return $d$
5      $p_1 := 2^n p(x/2)$; $p_2 := \mathbf{p_1}(\mathbf{x}+\mathbf{1})$
6      if $x \mid p_2$ then $m := 1$ else $m := 0$
7      $m' :=$ NumberInZeroOne($p_1$)
8      $m = m +$ NumberInZeroOne($p_2$)
9      return $m + m'$
10 end
```

Reformulate the problem: Pascal's triangle

Example

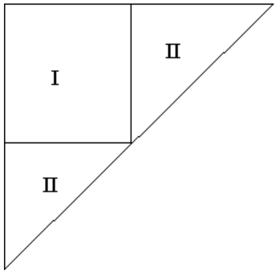
For $f(x) = a_3x^3 + a_2x^2 + a_1x + a_0$, we have

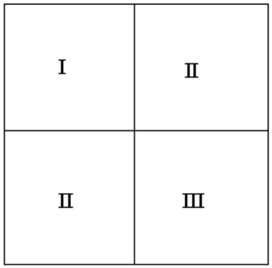
$$f(x+1) = a_3x^3 + (a_2 + 3a_3)x^2 + (a_1 + 2a_2 + 3a_3)x + (a_0 + a_1 + a_2 + a_3)$$

That is:

$$\begin{tikzcd}
 & 0 & & 0 & & 0 & & 0 & \\
a_3 & + & & + & & + & & + \arrow[r] & c_3 \\
a_2 & + & & + & & + \arrow[r] & c_2 & & \\
a_1 & + & & + \arrow[r] & c_1 & \searrow & & & \\
a_0 & + \arrow[r] & c_0 & & & & & &
\end{tikzcd}$$

Work, span and parallelism





For Tableau, we have

- **work:** $U_1(n) = 4U_1(n/2) + 1$, so $U_1(n) = \Theta(n^2)$.
- **span:** $U_\infty(n) = 3U_\infty(n/2) + 1$, so $U_\infty(n) = \Theta(n^{log_2 3})$.

For Pascal's triangle, we have

- **work:** $T_1(n) = 2T_1(n/2) + U_1(n/2)$, so $T_1(n) = \Theta(n^2)$.
- **span:** $T_\infty(n) = T_\infty(n/2) + U_\infty(n/2)$, so $T_\infty(n) = \Theta(n^{log_2 3})$.

The parallelism for both is $\Theta(n^{0.45})$.

Space and cache complexity

Space complexity

Since only the coefficients of $f(x+1)$ matter, computations can be done in place, so $\Theta(n)$.

Cache complexity

For two-way Tableau, we have

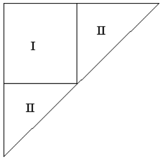
$$Q(n) = \begin{cases} 2n/L + 2 & n \leq \alpha Z \\ 4Q(n/2) + 1 & \text{otherwise} \end{cases} \quad \text{thus} \quad Q(n) = \Theta(n^2/ZL)$$

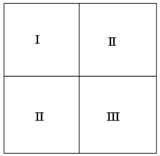
Then for the Pascal's triangle:

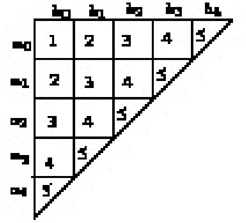
$$Q(n) = \begin{cases} 2n/L + 2 & n \leq \alpha Z \\ 2Q(n/2) + \Theta(n^2/ZL) & \text{otherwise} \end{cases} \quad \text{thus} \quad Q(n) = \Theta(n^2/ZL)$$

Using the Hong-Kung lower bound one can prove that this is optimal.

Increasing the parallelism







Using a k-way divide and conquer

Yes, but the cache complexity then depends linearly on k^2.

Using a blocking strategy

One can partition the entire Pascal Triangle into $B \times B$ blocks. Of course B should be tuned in order for a block to fit in cache.

Span and parallelism are now $\Theta(Bn)$ and $\Theta(n/B)$ respectively.

In addition, if B is well chosen, cache complexity remains optimal..

Experimental results

Table 1. Taylor shift (timings in seconds).

n	k	B	method	Bnd			Cnd			Random		
$\times 10^3$	$\times 10^3$			8p	1p	Sp	8p	1p	Sp	8p	1p	Sp
5	5	50	block	1.3	6.5	4.9	0.92	2.3	2.5	1.3	6.5	4.9
5	5	8	d-n-c	1.5	6.6	4.6	0.94	2.3	2.5	1.5	6.63	4.6
10	10	50	block	7.7	50.8	6.6	4.4	17.5	4.0	7.8	50.78	6.5
10	10	8	d-n-c	8.5	51.7	6.0	4.2	17.6	4.2	8.5	51.65	6.1
25	25	50	block	104	779	7.5	43	261	6.1	104	778.7	7.5
25	25	8	d-n-c	110	790	7.2	42	262	6.3	110	789.7	7.2

This machine has 8 GB memory and 6144 KB of L2 cache.
Each processor is Intel Xeon X5460 @3.16 GHz.
In the table, n and k denote the degree and coefficient size (number of bits) of the input polynomials.

Summary and notes

- The real roots of our polynomial (from the Hilbert 16 problem) with degree 852 and $6,355,573$ character long could be isolated on a 32-core node with 128 GB memory in 15 minutes.
- The implementation is in Cilk++.
- Work in progress includes the use of **dynamically sized blocks** to take into account the increase of work per block.
- Joint work with Changbo Chen and Yuzhen Xie.

Plan

Background

Background

- FFTs over finite fields is at the core of asymptotically fast polynomial arithmetic.
- Most FFTs on GPUs are for complex numbers, such as NVIDIA CUFFT library.

Testing in GB/s

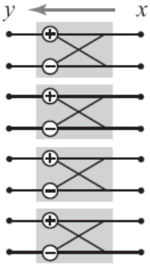
$\log_2 n$	memset	Main Mem to GPU	GPU to Main Mem	GPU Kernel
23	1.56	1.33	1.52	61.6
24	1.56	1.34	1.52	69.9
25	1.39	1.35	1.53	75.0
26	1.39	1.28	1.50	77.4
27	1.43	1.35	1.49	79.0

- Intel Core 2 Quad Q9400 @ 2.66GHz, 6GB memory, memory interface width 128 bits
- GeForce GTX 285, 1GB global memory, 30×8 cores, memory interface width 512 bits

Extract parallelism from structural formulas

$I_n \otimes A$: block parallelism

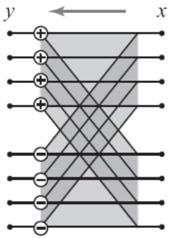
$$I_4 \otimes \mathrm{DFT}_2 = \begin{bmatrix} 1 & 1 & & & & & & \\ 1 & -1 & & & & & & \\ & & 1 & 1 & & & & \\ & & 1 & -1 & & & & \\ & & & & 1 & 1 & & \\ & & & & 1 & -1 & & \\ & & & & & & 1 & 1 \\ & & & & & & 1 & -1 \end{bmatrix}$$



Extract parallelism from structural formulas

$A \otimes I_n$: vector parallelism

$$\mathrm{DFT}_2 \otimes I_4 = \begin{bmatrix} 1 & & & & 1 & & & \\ & 1 & & & & 1 & & \\ & & 1 & & & & 1 & \\ & & & 1 & & & & 1 \\ 1 & & & & -1 & & & \\ & 1 & & & & -1 & & \\ & & 1 & & & & -1 & \\ & & & 1 & & & & -1 \end{bmatrix}$$



Stockham FFT

$$\mathrm{DFT}_{2^k} = \prod_{i=0}^{k-1} \underbrace{(\mathrm{DFT}_2 \otimes I_{2^{k-1}})}_{\text{butterfly}} \underbrace{(D_{2,2^{k-i-1}} \otimes I_{2^i})}_{\text{twiddling}} \underbrace{(L_2^{2^{k-i}} \otimes I_{2^i})}_{\text{reordering}}$$

```
void stockham_dev(int *X_d, int n, int k, const int *W_d, int p)
{
  int *Y_d;
  cudaMalloc((void **)&Y_d, sizeof(int) * n);
  butterfly_dev(Y_d, X_d, k, p);
  for (int i = k - 2; i >= 0; --i) {
    stride_transpose2_dev(X_d, Y_d, k, i);
    stride_twiddle2_dev(X_d, W_d, k, i, p);
    butterfly_dev(Y_d, X_d, k, p);
  }
  cudaMemcpy(X_d, Y_d, sizeof(int)*n, cudaMemcpyDeviceToDevice);
  cudaFree(Y_d);
}
```

Cooley-Tukey FFT

$$\mathrm{DFT}_{2^k} = \left(\prod_{i=1}^{k} (I_{2^{i-1}} \otimes \mathrm{DFT}_2 \otimes I_{2^{k-i}})\, T_{n,i} \right) R_n$$

with the twiddle factor matrix $T_{n,i} = I_{2^{i-1}} \otimes D_{2,2^{k-i}}$ and the bit-reversal permutation matrix

$$R_n = (I_{n/2} \otimes L_2^2)(I_{n/2^2} \otimes L_2^4) \cdots (I_1 \otimes L_2^n).$$

Timing FFT in milliseconds

e	modpn	Cooley-Tukey		C-T + Mem		Stockham		S + Mem	
		time	ratio	time	ratio	time	ratio	time	ratio
12	1	1	1.0	1	1.0	2	0.5	2	0.5
13	1	2	0.5	2	0.5	2	0.5	3	0.3
14	3	1	3.0	2	1.5	2	1.5	3	1.0
15	4	2	2.0	2	2.0	3	2.0	3	1.3
16	10	3	3.3	3	3.3	3	3.3	4	3.3
17	16	4	4.0	5	3.2	3	5.3	5	3.2
18	37	6	6.2	9	4.1	4	9.3	7	5.3
19	71	11	6.5	15	6.5	6	11.8	10	7.1
20	174	22	7.9	28	6.2	9	19.3	16	10.9
21	470	44	10.7	56	8.4	16	29.4	28	16.8
22	997	83	12.0	105	9.5	29	34.4	52	19.2
23	2070	165	12.5	210	9.9	56	37.0	101	20.5
24	4194	330	12.7	418	10.0	113	37.0	201	20.9
25	8611	667	12.9	842	10.2	230	37.4	405	21.2
26	17617	1338	13.2	1686	10.4	473	37.2	822	21.4

The GPU is GTX 285.

Solving polynomial systems with GPU support

Main idea

Solving $P(x, y) = Q(x, y) = 0$ is essentially done as follows:

1. Determine necessary conditions on x for $P(x)(y)$ and $Q(x)(y)$ to have common roots; such x's are roots of the **resultant** $R(x)$ of P, Q w.r.t. y.
2. For $x = x_0$ such that x_0 is a root of R determine the common solutions of $P(x_0)(y) = 0$ and $Q(x_0)(y) = 0$; this is essentially a GCD computation.

Both steps can be easily deduced from a so-called **Subresultant Chain Computation**

Subresultant chain computation

$$\begin{tikzcd}
P,\ Q \text{ in } \mathbb{Z}_p[x_1,\ldots,x_n,y] \arrow[rr, "\text{Direct computation}"] \arrow[d, "\text{Kronecker's substitution}"] & & \mathsf{subres}(P,Q,y) \in \mathbb{Z}_p[x_1,\ldots,x_n,y] \\
F,\ G \text{ in } \mathbb{Z}_p[x,y] \arrow[d, "\text{Random translation } \phi_a"] & & \mathsf{subres}(F(x,y),G(x,y),y) \arrow[u, "\text{Inverse Kronecker}"'] \\
F',\ G' \text{ in } \mathbb{Z}_p[x,y] \arrow[r] \arrow[d, "\text{FFT}"] & \text{exit, if several random choices } a \text{ failed} & \mathsf{subres}(F'(x,y),G'(x,y),y) \arrow[u, "\text{Inverse } \phi_a"'] \\
F'(\omega^i,y),\ G'(\omega^i,y) \text{ in } \mathbb{Z}_p[y] \arrow[rr, "\text{Brown's algorithm}"] & & \mathsf{subres}(F'(\omega^i,y),G'(\omega^i,y),y) \arrow[u, "\text{Inverse FFT}"']
\end{tikzcd}$$

Subresultant chain by evaluation/interpolation

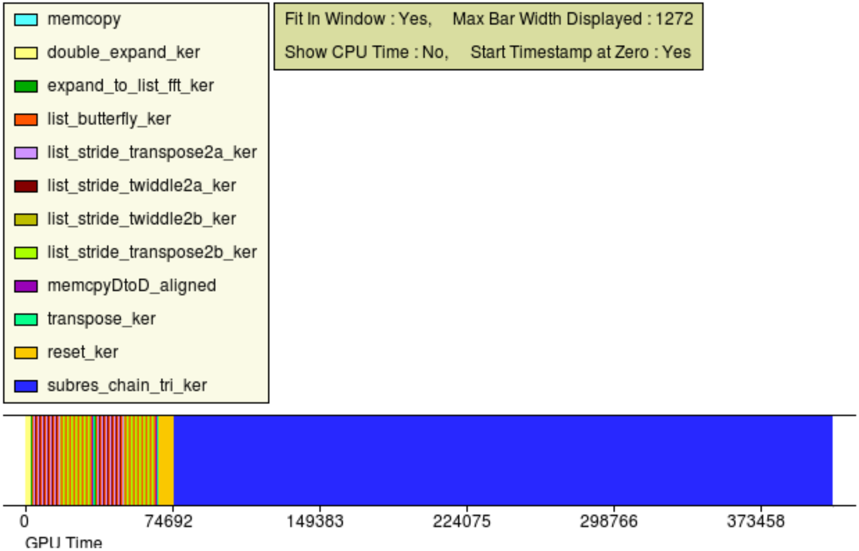
Different Strategies

- FFT based techniques
  - Fourier prime limitation,
  - a valid grid is required,
  - translations are tried a few times.
- Subproduct tree techniques

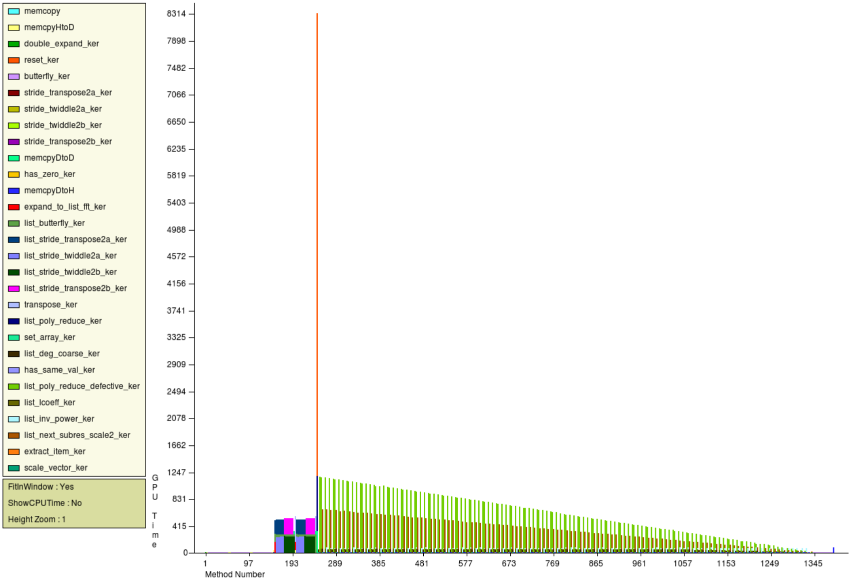
FFT scube on the GPU

- Coarse-grained construction (always works)
- Fine-grained construction (requires some genericity assumption)

Profiling coarse-grained implementation



Profiling fine-grained implementation



Computing resultants

d	t_0	t_1	t_1/t_0
30	0.23	0.29	1.3
40	0.23	0.43	1.9
50	0.27	1.14	4.2
60	0.27	1.53	5.7
70	0.31	3.95	12.7
80	0.32	4.88	15.3
90	0.35	5.95	17.0
100	0.50	19.10	38.2
110	0.53	17.89	33.8
120	0.58	19.72	34.0

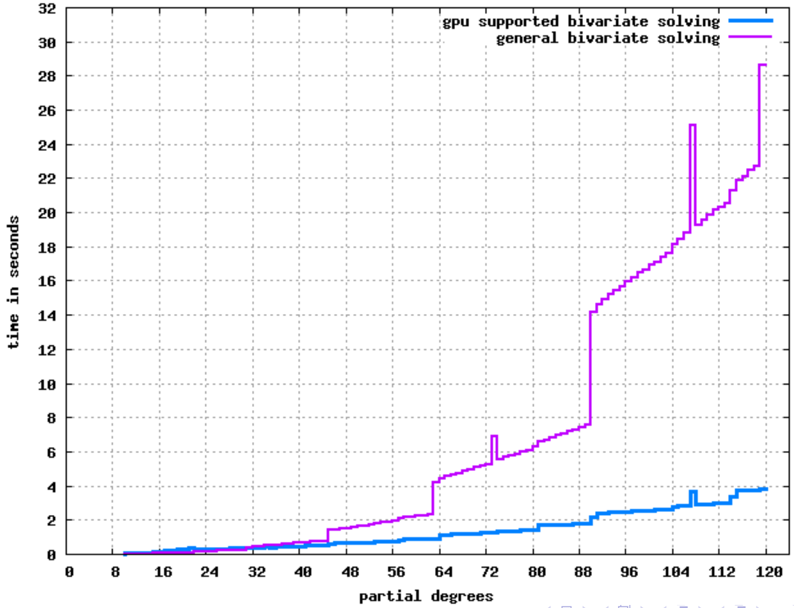
Bivariate dense polynomials of total degree d.

d	t_0	t_1	t_1/t_0
8	0.23	0.76	3.3
9	0.24	0.85	3.5
10	0.25	0.98	3.9
11	0.24	1.10	4.6
12	0.30	4.96	16.5
13	0.31	5.52	17.8
14	0.32	6.07	19.0
15	0.78	8.95	11.5
16	0.65	31.65	48.7
17	0.66	34.55	52.3
18	3.46	47.54	13.7
19	0.73	51.04	69.9
20	0.75	43.12	57.5

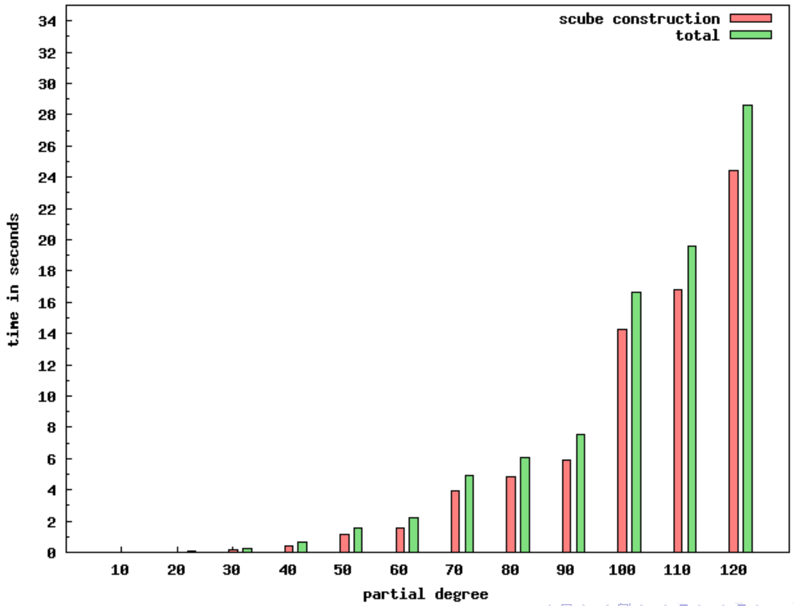
Trivariate dense polynomials of total degree d.

- t_0, GPU fft code
- t_1, CPU fft code
- Nvidia Tesla C2050

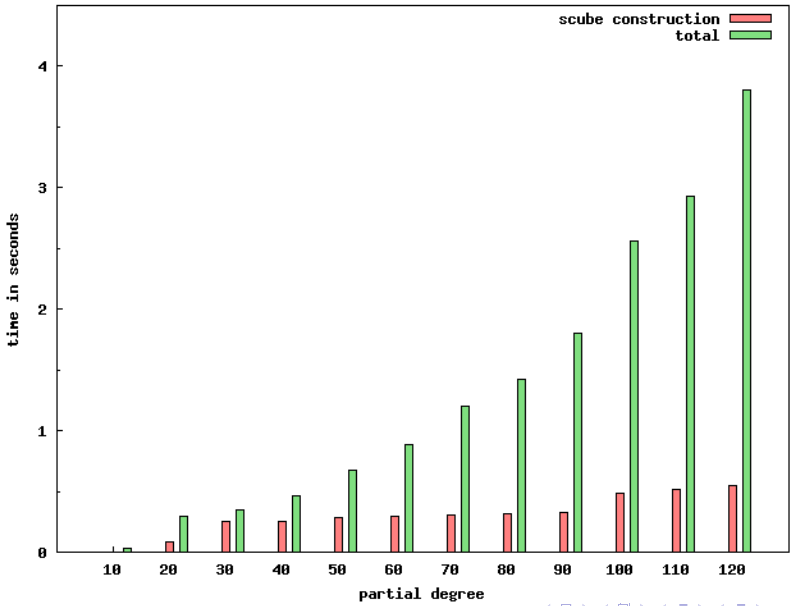
Bivariate solver



Bivariate solver on the CPU



Bivariate solver on the GPU



Solving bivariate systems in seconds

d	t_0(gpu)	t_1(total)	t_2 (cpu)	t_3 (total)	t_2/t_0	t_3/t_1
30	0.25	0.35	0.14	0.25	0.6	0.7
40	0.25	0.46	0.42	0.64	1.7	1.4
50	0.28	0.67	1.14	1.56	4.1	2.3
60	0.29	0.88	1.54	2.20	5.3	2.5
70	0.31	1.20	3.94	4.94	12.7	4.1
80	0.32	1.42	4.84	6.06	15.1	4.3
90	0.33	1.80	5.94	7.54	18.0	4.2
100	0.48	2.56	14.23	16.66	29.7	6.5
110	0.52	2.93	16.78	19.58	32.1	6.7
120	0.55	3.80	24.41	28.60	44.4	7.5

- d : total degree of the input polynomial
- t_0 : GPU FFT based scube construction
- t_1 : total time for solving with GPU code
- t_2 : CPU FFT based scube construction
- t_3 : total time for solving without CPU code

Summary and notes

- The Stockham FFT achieves a speedup factor of 21 for large FFT degrees, comparing to the `modpn` serial implementation.
- The subresultant chain construction has been improved by a factor of (up to) 44 on the GPU.
- For the bivariate solver, more code has to be ported to GPU (mainly univariate polynomial GCDs)
- Nevertheless the GPU-based code solves within a second, polynomial systems for which pure serial code takes 7.5 sec.
- The goal is to make bivariate and trivariate system solvers as fast as a univariate GCD routine in MAPLE.
- Our H 16 cluster will rely heavily on those!
- Joint work with Wei Pan.

The predator-prey biological model (1/4)

- Two species interact, one is a predator and one is its prey, according to the pair of differential equations:

$$\begin{cases} \frac{dx}{dt} & = & x(a - by) \\ \frac{dy}{dt} & = & -y(c - dx). \end{cases}$$

- Say x and y are numbers of **carnivores** and **herbivores**, while a, b, c, d are parameters.

- Population equilibria at:

$$\begin{cases} x(a - by) = 0 \\ y(c - dx) = 0. \end{cases}$$

- This gives two solutions:

$$(x, y) = (0, 0) \text{ and } (x, y) = (\frac{c}{d}, \frac{b}{a}).$$

The predator-prey biological model (2/4)

- Stability analysis of the hyperbolic equilibria via linearization (Hartman and Grobman Theorem). The Jacobian matrix of the system:

$$J(x,y) = \begin{bmatrix} a - by & -bx \\ dy & dx - c \end{bmatrix}.$$

- Its characteristic polynomial is:

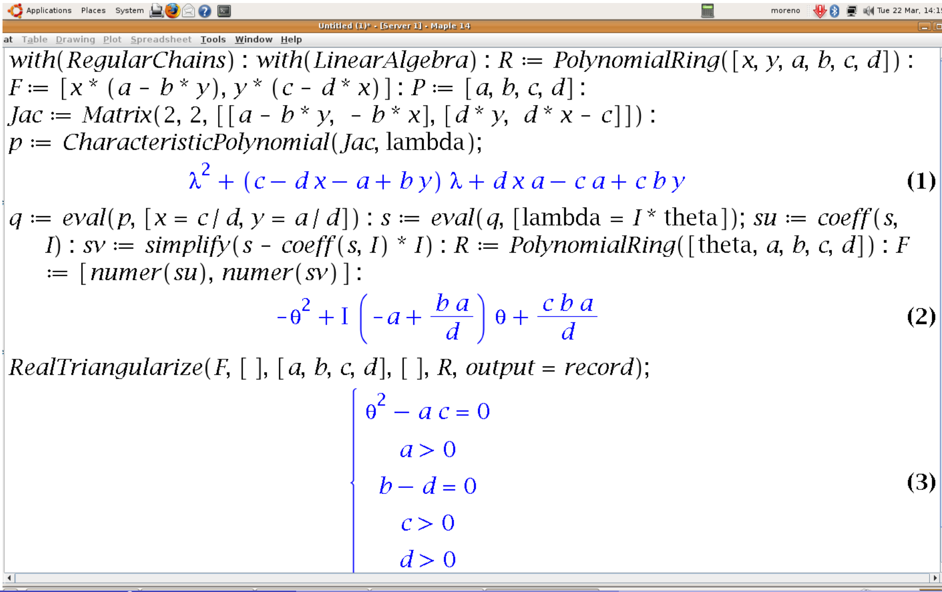
$$p = \lambda^2 + (c - xd - a + yb)\lambda + xad - ac + ybc.$$

- At $(x, y) = (0, 0)$, we have a saddle point, thus instable, since:

$$p = -(\lambda + c)(-\lambda + a)$$

- At $(x, y) = (\frac{c}{d}, \frac{b}{a})$, the characteristic polynomial p has roots with zero real part, as we shall see.

The predator-prey biological model (3/4)


Applications Places System
moreno Tue 22 Mar, 14:1
Untitled (1)* - [Server 1] - Maple 14
at Table Drawing Plot Spreadsheet Tools Window Help

```
with(RegularChains) : with(LinearAlgebra) : R := PolynomialRing([x, y, a, b, c, d]) :
F := [x * (a - b * y), y * (c - d * x)] : P := [a, b, c, d] :
Jac := Matrix(2, 2, [[a - b * y,  - b * x], [d * y,  d * x - c]]) :
p := CharacteristicPolynomial(Jac, lambda);
```

$$\lambda^2 + (c - d\,x - a + b\,y)\,\lambda + d\,x\,a - c\,a + c\,b\,y \qquad (1)$$

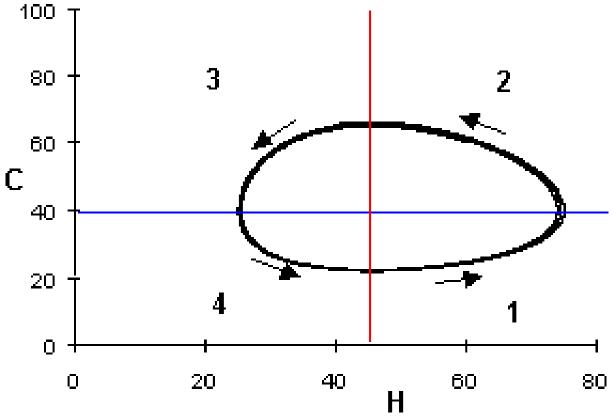
```
q := eval(p, [x = c / d, y = a / d]) : s := eval(q, [lambda = I * theta]); su := coeff(s,
    I) : sv := simplify(s - coeff(s, I) * I) : R := PolynomialRing([theta, a, b, c, d]) : F
    := [numer(su), numer(sv)] :
```

$$-\theta^2 + \mathrm{I}\left(-a + \frac{b\,a}{d}\right)\theta + \frac{c\,b\,a}{d} \qquad (2)$$

```
RealTriangularize(F, [ ], [a, b, c, d], [ ], R, output = record);
```

$$\left\{\begin{array}{l} \theta^2 - a\,c = 0 \\ a > 0 \\ b - d = 0 \\ c > 0 \\ d > 0 \end{array}\right. \qquad (3)$$

The predator-prey biological model (4/4)



Therefore, at $(x, y) = (\frac{c}{d}, \frac{b}{a})$, we have a limit cycle:

- as the number of herbivores increases, then so does that of carnivores.
- but as that of carnivores increases, that of herbivores decreases, . . .