— Solving Bivariate Polynomial Systems on a GPU —

Marc Moreno Maza and Wei Pan

ORCCA Ontario Research Centre for Computer Algebra

Western

Overview

We present a CUDA implementation of dense multivariate polynomial arithmetic based on Fast Fourier Transforms (FFT) over finite fields. The motivation of our work is to support polynomial system solvers based on the notion of a *regular chains* [3] where the core algorithms [1] rely on the theory of polynomial subresultants.

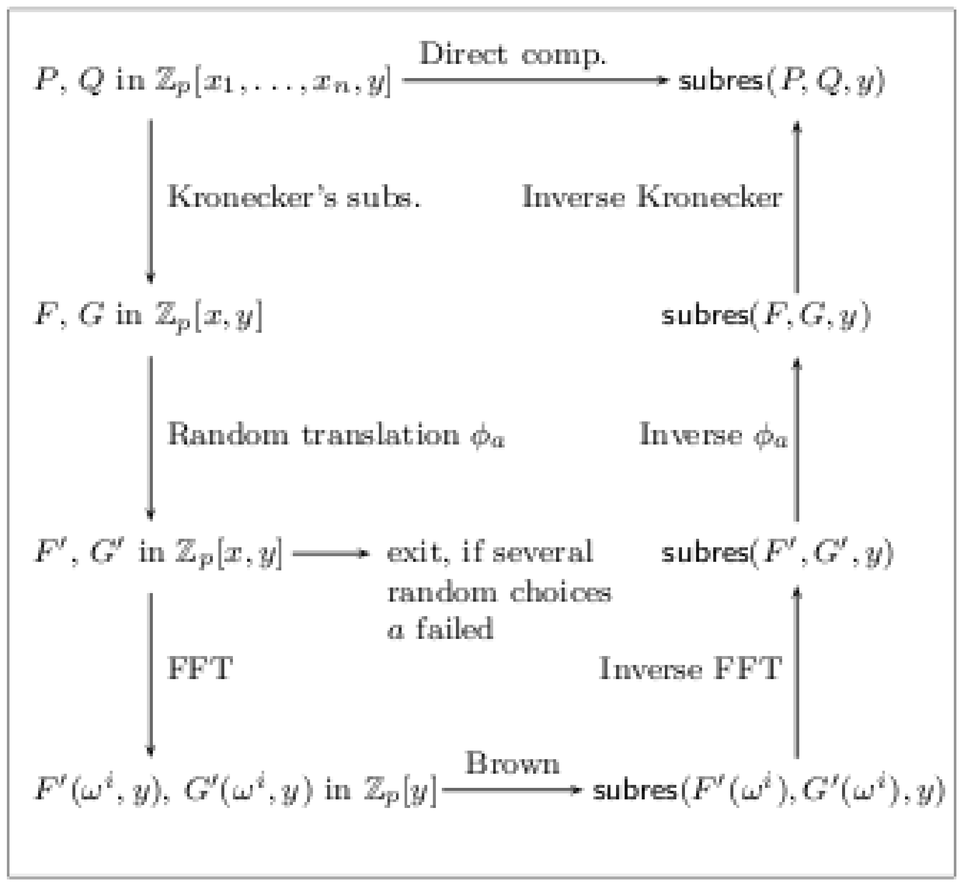
In [4], we have shown that the dominant cost of those algorithms can be essentially reduced to that of subresultant chain computations.

Methodology

Following an idea proposed by Collins in [2], we compute subresultant chains by evaluation-interpolation. Moreover, we rely on FFT techniques: once the input polynomials are sampled on a so-called *FFT grid*, we employ Brown's Algorithm to compute the polynomial subresultants at each sample.

All these computations are performed on the device for which we have designed two specific implementations of Brown's Algorithm. The first one called *fine grained* requires that every sampled subresultant chain has the same degree sequence of non-zero subresultants. This approach creates more parallelism but fails if its assumption does not hold. When this happens a *coarse grained* approach does the job without any assumption but with reduced performance.

The sampled subresultant chains can be exploited by a high-level algorithm running on the host. A simple case is that of resultant computation. A more advanced one is that of the bivariate solver algorithm of [5].



For efficiency reasons (locality, parallelism, ...) the case of n-variate polynomials with $n > 3$ is reduced to the bivariate case via Kronecker's substitution. Another feature of our implementation is the use of random translations of the form $\phi_a : x \longmapsto x + a$ (applied to the polynomials $F', G' \in \mathbb{z}_p[x, y]$, in order to find a valid FFT grid, that is, a FFT grid at every point of which $\mathrm{lc}(F', y)$ and $\mathrm{lc}(G', y)$ do not vanish.

Acknowledgements

NSERC CRSNG

Maplesoft Mathematics • Modeling • Simulation

MITACS

Experimental results

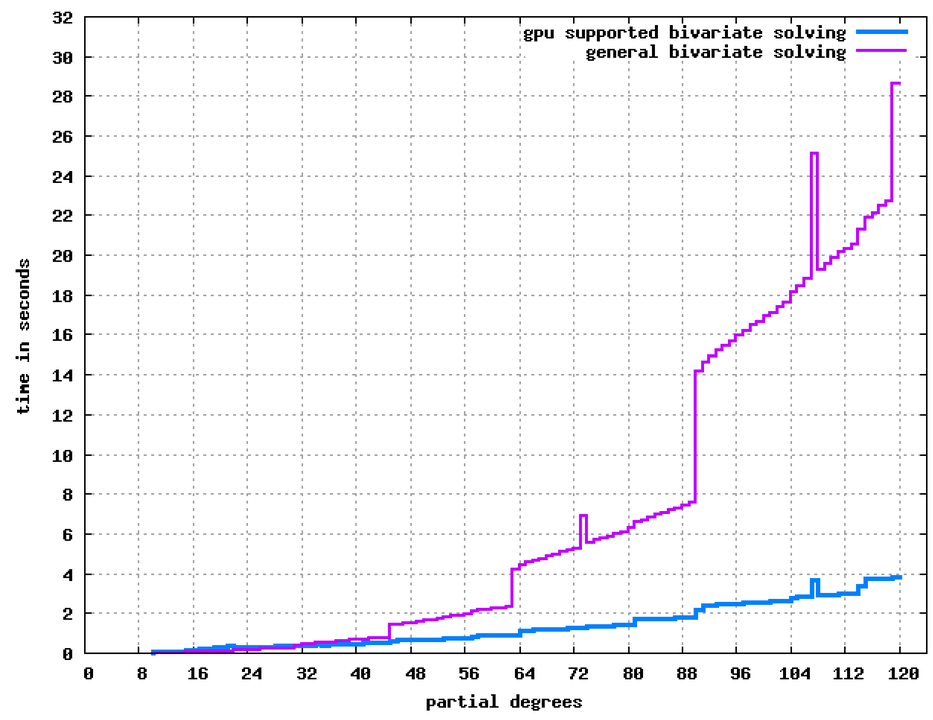
The table below lists our experimental results for computing resultants for trivariate random dense polynomials in x, y, z.

The first column shows the common partial degree d in x, y and z. The second one, labelled by t_0, is the timing for GPU supported code, which includes host-device data transfers The third column, labelled by t_1, shows our pure CPU serial C code in the `modpn` library [5].

d	t_0	t_1	t_1/t_0
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

Note that all but $d = 15$ and $d = 18$ employ the fine-grained version of our implementations of Brown's Algorithm. When the coarse-grained method is forced to be used, the speedup it achieves drops significantly. These data were obtained with an NVIDIA card GTX 285.

For our bivariate system solving benchmarks, the input polynomials are dense with coefficients modulo $p = 469762049$. The figure below shows the comparison of the running time with and without GPU acceleration. The largest speedup we achieve is approximately 7.5, since univariate polynomials GCDs are still computed on the host! These data were obtained with an NVIDIA card Tesla 2050.



References

[1] C. Chen and M. Moreno Maza. Algorithms for computing triangular decompositions of polynomial systems. In *proceedings of ISSAC 2011*. ACM Press, 2011.

[2] G.E. Collins. The calculation of multivariate polynomial resultants. *Journal of the ACM*, 18(4):515–532, 1971.

[3] M. Kalkbrener. A generalized Euclidean algorithm for computing triangular representations of algebraic varieties. *J. Symb. Comp.*, 15:143–167, 1993.

[4] X. Li, M. Moreno Maza, and W. Pan. Computations modulo regular chains. In *Proc. ISSAC'09*, pages 239–246, New York, NY, USA, 2009. ACM Press.

[5] X. Li, M. Moreno Maza, R. Rasheed, and É. Schost. The modpn library: Bringing fast polynomial arithmetic into maple, 2008. To appear in the J. of Symbolic Computation.