Journal of Symbolic Computation 79 (2017) 535–558

Contents lists available at ScienceDirect

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Parallel degree computation for binomial systems

Tianran Chen [a], Dhagash Mehta [b,c]

[a] Department of Mathematics, Michigan State University, East Lansing, MI, USA
[b] Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, Notre Dame, IN 46545, USA
[c] Centre for the Subatomic Structure of Matter, Department of Physics, School of Physical Sciences, University of Adelaide, Adelaide, South Australia 5005, Australia

ARTICLE INFO

Article history:
Received 1 July 2015
Accepted 19 August 2015
Available online 16 July 2016

Keywords:
Binomial systems
Homotopy continuation
Algebraic geometry
BKK root-count
GPU computing
Supersymmetric gauge theories

ABSTRACT

Solution sets of systems of binomial equations are of great interest in applied mathematics. For both theoretic and applied purposes, the degree of a solution set (its maximum number of isolated intersections with an affine space of complementary dimension) often plays an important role in understanding its geometric structure. This paper proposes a specialized parallel algorithm for computing the degree on GPUs that takes advantage of the massively parallel nature of GPU devices. The preliminary implementation shows remarkable efficiency and scalability when compared to the closest CPU-based counterpart. As a case study, the algorithm is applied to the master space problem of $\mathcal{N} = 1$ gauge theories. The GPU-based implementation achieves nearly 30 fold speedup over its CPU-only counterpart enabling the discovery of previously unknown results.

1. Introduction

The problem of solving a system of polynomial equations is an important problem in computational mathematics. Of special interest is the problem of solving systems of binomial equations (or simply binomial systems) for they appear naturally in many applications and specialized algorithms exist (e.g., Kahle, 2010). In many applications, only the solutions of a binomial system for

E-mail addresses: chentia1@msu.edu (T. Chen), dmehta@nd.edu (D. Mehta).

http://dx.doi.org/10.1016/j.jsc.2016.07.018
0747-7171/

which no variable is zero are needed. That is, we are only interested in solutions inside $(\mathbb{C}^*)^n$ where $\mathbb{C}^* = \mathbb{C} \setminus \{0\}$. The collection of such solutions is known as the $\mathbb{C}^*$-solution set and will be the focus of this article. Such a $\mathbb{C}^*$-solution set of a binomial system has a well defined dimension and can be decomposed into "irreducible components". Of particular interest in the context of the present contribution are the components of positive dimension (i.e., nonisolated solutions). The *degree* of each component of positive dimension is the maximum number of points in the intersection of the component with an affine linear space of complementary dimension.

Based on a class of algorithms for the related problem of "mixed volume computation" the present article proposes a specialized numerical parallel algorithm for computing the degree of a $\mathbb{C}^*$-solution set of a system of binomial equations. The novel features of this algorithm include

- Specially designed for GPU (graphics processing unit) devices, this algorithm takes advantage of the massively parallel nature of GPUs;
- Utilizing the special structure of binomial systems, this algorithm employs techniques that are not available (or not known to be available) for the more general "mixed volume computation" which greatly improves the overall efficiency;
- Based on a formulation of the degree as the volume of a lattice polytope, the algorithm computes the degree without using any sample points of the solution set (e.g. "witness sets" in Numerical Algebraic Geometry).

As a case study, we applied this algorithm to a family of binomial systems coming from particle physics, called the master space of $\mathcal{N} = 1$ gauge theories. The preliminary implementation for GPU devices built on top of the NVidia CUDA framework has already shown promising results. In particular, it outperforms existing CPU-based programs in both absolute efficiency and parallel scalability by a large margin, and it produced previously unknown results. Remarkably, with multiple GPU devices (on the same computer), the GPU based implementation exhibits much better performance, in many cases, than small to medium sized computer clusters.

Though the main focus of this article is the parallel algorithm that directly computes the degree of the $\mathbb{C}^*$-solution set of a binomial system without using any sample points (e.g. "witness points"), this algorithm actually generates, as by-products, information from which "witness sets" of each component can be computed indirectly via a specialized homotopy continuation method. Developed in Sommese and Wampler (1996) as the numerical representations of irreducible components of algebraic sets, "witness sets" are, in a sense, the fundamental data structure in the field of Numerical Algebraic Geometry (Sommese and Wampler, 1996). The secondary contribution of this article is an indirect homotopy-based method for computing the "witness set" of each irreducible component of a positive dimensional $\mathbb{C}^*$-solution set of a binomial system. The potential benefit of this approach is that the number of homotopy paths required to compute a witness set (of a component) is exactly the number of witness points in the set. That is, each homotopy path yields a unique witness point and no paths diverge.

This article is structured as follows: First, necessary notation and concepts are introduced. In particular, we shall review basic geometric properties of the $\mathbb{C}^*$-solution set defined by a binomial system. Then in §3 we propose a highly scalable parallel algorithm for computing the degree of the $\mathbb{C}^*$-solution set of a binomial system. Aiming to be self contained, the algorithm is presented without direct reference to related concepts in "mixed volume computation". In §5, however, this close tie is discussed in detail. The construction of the specialized homotopies for computing witness sets is described in §4. The case study, of the master space of $\mathcal{N} = 1$ gauge theories, arising from string theory, is presented in §6, and we show the previously unknown results obtained by the proposed parallel algorithm. Certain technical but necessary results are included in the appendix for completeness.

2. Laurent binomial systems and their solution sets

We first review necessary concepts and notation. For positive integers m and n, let $M_{n\times m}(\mathbb{Z})$ denote the set of all $n \times m$ integer matrices. A square integer matrix is said to be **unimodular** if its determinant is ± 1. Note that such a square matrix $A \in M_{n\times n}(\mathbb{Z})$ has a unique inverse $A^{-1} = \frac{1}{\det A} \operatorname{adj} A$

that is also in $M_{n\times n}(\mathbb{Z})$, where adj A is the adjugate matrix of A. The $n\times n$ identity matrix in $M_{n\times n}(\mathbb{Z})$ is denoted by I_n.

The theoretical tools involved in this article are more naturally developed in the context of the more general "Laurent binomial systems" where negative exponents are allowed. For variables $\boldsymbol{x} = (x_1, \ldots, x_n)$, a **Laurent monomial** in $\boldsymbol{x}$ is an expression of the form $x_1^{a_1}\cdots x_n^{a_n}$ where $a_1, \ldots, a_n \in \mathbb{Z}$ (which may be zero or negative). For convenience, we shall write $\boldsymbol{a} = (a_1, \ldots, a_n)^\top \in \mathbb{Z}^n$ and use the "vector exponent" notation

$$\boldsymbol{x}^{\boldsymbol{a}} = (x_1, \ldots, x_n)^{\begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}} = x_1^{a_1} \cdots x_n^{a_n}$$

to denote a Laurent monomial. Similarly, for an integer matrix $A \in M_{n\times m}(\mathbb{Z})$ with columns $\boldsymbol{a}^{(1)}, \ldots, \boldsymbol{a}^{(m)} \in \mathbb{Z}^n$, the "matrix exponent" notation will be used for an m-tuple of Laurent monomials:

$$\boldsymbol{x}^A = \boldsymbol{x}^{\begin{pmatrix} \boldsymbol{a}^{(1)} & \cdots & \boldsymbol{a}^{(m)} \end{pmatrix}} := (\boldsymbol{x}^{\boldsymbol{a}^{(1)}}, \ldots, \boldsymbol{x}^{\boldsymbol{a}^{(m)}}). \tag{1}$$

This notation is particularly convenient since the familiar identities $\boldsymbol{x}^{I_n} = \boldsymbol{x}$ and $(\boldsymbol{x}^A)^B = \boldsymbol{x}^{AB}$ still hold. Each matrix $A \in M_{n\times m}(\mathbb{Z})$ induces a function from $(\mathbb{C}^*)^n$ to $(\mathbb{C}^*)^m$ given by $\boldsymbol{x} \mapsto \boldsymbol{x}^A$. Of particular importance is the function induced by a unimodular matrix $A \in M_{n\times n}(\mathbb{Z})$ since A^{-1} is also in $M_{n\times n}(\mathbb{Z})$, and hence functions $\boldsymbol{x} \mapsto \boldsymbol{x}^A$ and $\boldsymbol{x} \mapsto \boldsymbol{x}^{A^{-1}}$ are the inverses of each other ($(\boldsymbol{x}^A)^{A^{-1}} = \boldsymbol{x}^{A\,A^{-1}} = \boldsymbol{x}^{I_n} = \boldsymbol{x}$).

A **Laurent binomial** is an expression of the form $c_1\boldsymbol{x}^{\boldsymbol{\alpha}} + c_2\boldsymbol{x}^{\boldsymbol{\beta}}$ for some $c_1, c_2 \in \mathbb{C}^*$ and $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{Z}^n$. This article focuses on the properties of the solution set of systems of Laurent binomials equations, or simply **Laurent binomial systems**, over $(\mathbb{C}^*)^n$. Stated formally, given exponent vectors $\boldsymbol{\alpha}^{(1)}, \ldots, \boldsymbol{\alpha}^{(m)}$, $\boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\beta}^{(m)} \in \mathbb{Z}^n$ and the coefficients $c_{i,j} \in \mathbb{C}^*$, the goal is to describe the set of all $\boldsymbol{x} \in (\mathbb{C}^*)^n$ that satisfies the system of equations

$$\begin{cases} c_{1,1}\boldsymbol{x}^{\boldsymbol{\alpha}^{(1)}} + c_{1,2}\boldsymbol{x}^{\boldsymbol{\beta}^{(1)}} = 0 \\ \qquad\qquad\qquad\qquad \vdots \\ c_{m,1}\boldsymbol{x}^{\boldsymbol{\alpha}^{(m)}} + c_{m,2}\boldsymbol{x}^{\boldsymbol{\beta}^{(m)}} = 0. \end{cases} \tag{2}$$

Since only the solutions in $(\mathbb{C}^*)^n$ are concerned, this system is clearly equivalent to

$$(\boldsymbol{x}^{\boldsymbol{\alpha}^{(1)}-\boldsymbol{\beta}^{(1)}}, \ldots, \boldsymbol{x}^{\boldsymbol{\alpha}^{(m)}-\boldsymbol{\beta}^{(m)}}) = (-c_{1,2}/c_{1,1}, \ldots, -c_{m,2}/c_{m,1}).$$

With the more compact "matrix exponent" notation in (1), this system can simply be written as

$$\boldsymbol{x}^A = \boldsymbol{b} \qquad \text{or equivalently} \qquad \boldsymbol{x}^A - \boldsymbol{b} = \boldsymbol{0} \tag{3}$$

where the integer matrix $A \in M_{n\times m}(\mathbb{Z})$, having columns $\boldsymbol{\alpha}^{(1)} - \boldsymbol{\beta}^{(1)}, \ldots, \boldsymbol{\alpha}^{(m)} - \boldsymbol{\beta}^{(m)}$, represents the exponents appeared in the Laurent monomials and the vector $\boldsymbol{b} = (-c_{1,2}/c_{1,1}, \ldots, -c_{m,2}/c_{m,1})^\top \in (\mathbb{C}^*)^m$ collects all the coefficients. The solution set of (3) over $(\mathbb{C}^*)^n$ shall be denoted by

$$\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b}) = \{\boldsymbol{x} \in (\mathbb{C}^*)^n \mid \boldsymbol{x}^A - \boldsymbol{b} = \boldsymbol{0}\}. \tag{4}$$

Note that $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$ is *quasi-projective* since it is the complement of the (complex) algebraic set $\mathbb{C}^n \setminus (\mathbb{C}^*)^n = \{(x_1, \ldots, x_n) \mid x_1 \cdots x_n = 0\}$ within the (complex) solution set of the original binomial system (2). In the context of algebraic geometry, such a set has a well defined *dimension* and *irreducible decomposition* (its decompositions as a union of "irreducible components"). Moreover, the concept of *degree* can be defined for each of its irreducible component (which depends on the defining equation $\boldsymbol{x}^A - \boldsymbol{b}$). In the following, we shall review the main structural theorem that ties these properties directly to the properties of the matrix A. Since the explicit "global parametrization" (to be described in Proposition 1) is needed in the formulation of the degree computation problem, a brief derivation of these results is therefore included. A more detailed summary can be found in the article

by Chen and Li (2014). In depth theoretical discussions can be found in standard references such as Cox et al. (2011), Eisenbud and Sturmfels (1996), Fulton (1993), Miller and Sturmfels (2005), Sturmfels (1997). Certain computational aspects have been studied in Kahle (2010), Kahle and Miller (2014).

An important tool in understanding the structure of $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$ is the *Smith Normal Form* (Smith, 1861) of the exponent matrix $A \in M_{n\times m}(\mathbb{Z})$: there are unimodular square matrices $P \in M_{n\times n}(\mathbb{Z})$ and $Q \in M_{m\times m}(\mathbb{Z})$ such that

$$P\,A\,Q \;=\; \begin{pmatrix} d_1 & & & & & \\ & \ddots & & & & \\ & & d_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix} \in M_{n\times m}(\mathbb{Z}) \tag{5}$$

with nonzero integers $d_1 \mid d_2 \mid \cdots \mid d_r$ for $r = \operatorname{rank} A$. Here, $a \mid b$ means a divides b as usual. The nonzero diagonal elements $d_1, \ldots, d_r$ are unique up to the signs and known as *invariant factors* of A. The transformation matrices P and Q, however, are not unique in general. Although they are to used in following structural description of $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$, the description itself is independent from the choices of P and Q. This subtle point will be clarified in Remark 4.

Let $P_r \in M_{r\times n}(\mathbb{Z})$ and $P_0 \in M_{(n-r)\times n}(\mathbb{Z})$ be the top r rows and the remaining $n-r$ rows of P in (5) respectively. Similarly, let $Q_r \in M_{m\times r}(\mathbb{Z})$ and $Q_0 \in M_{m\times(m-r)}(\mathbb{Z})$ be the left r columns and the remaining $m-r$ columns of Q respectively. With these notations, the Smith Normal Form of (5) of A can be written as

$$\begin{pmatrix} P_r \\ P_0 \end{pmatrix} A \begin{pmatrix} Q_r & Q_0 \end{pmatrix} = \begin{pmatrix} D & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{6}$$

with $D = \operatorname{diag}(d_1, \ldots, d_r) \in M_{r\times r}(\mathbb{Z})$ and $\mathbf{0}$'s representing zero block matrices of appropriate sizes. With this we can transform the binomial system $\boldsymbol{x}^A = \boldsymbol{b}$ into a form from which the structure of the $\mathbb{C}^*$-solution set can be easily extracted.

Since P and Q are both unimodular the maps $\boldsymbol{z} \mapsto \boldsymbol{z}^P$ and $\boldsymbol{y} \mapsto \boldsymbol{y}^Q$ are bijections on $(\mathbb{C}^*)^n$ and $(\mathbb{C}^*)^m$ respectively. Therefore, as far as the $\mathbb{C}^*$-solution set is concerned, the original system $\boldsymbol{x}^A = \boldsymbol{b}$ is equivalent to $(\boldsymbol{x}^A)^Q = \boldsymbol{x}^{A\,Q} = \boldsymbol{b}^Q$. Similarly, the solution set remains equivalent after the change of variables $\boldsymbol{x} = \boldsymbol{z}^P$, which produces

$$(\boldsymbol{z}^P)^{A\,Q} \;=\; \boldsymbol{z}^{P\,A\,Q} \;=\boldsymbol{z}^{\left(\begin{smallmatrix} D & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right)} \;=\; (\boldsymbol{z}^{\binom{D}{\mathbf{0}}}, \boldsymbol{z}^{\binom{\mathbf{0}}{\mathbf{0}}}) \;=\; \boldsymbol{b}^Q \;=\; (\boldsymbol{b}^{Q_r}, \boldsymbol{b}^{Q_0})\,.$$

Since $D = \operatorname{diag}(d_1, \ldots, d_r) \in M_{r\times r}(\mathbb{Z})$, this system can be decomposed into a combined system

$$(z_1^{d_1}, \ldots,\; z_r^{d_r}) = \boldsymbol{b}^{Q_r} \tag{7}$$

$$\mathbf{1} \;=\; \boldsymbol{b}^{Q_0} \tag{8}$$

$$z_{r+1}, \ldots, z_n : \;\text{free} \tag{9}$$

where (8) appears when $r < m$ with $\mathbf{1} = (1, \ldots, 1) \in (\mathbb{C}^*)^{m-r}$, and (9) appears when $r < n$. The word "free" in (9) means the system imposes no constraints on the $n-r$ variables $z_{r+1}, \ldots, z_n$. It is clear that if $r < m$, then the system is inconsistent unless $\mathbf{1} = \boldsymbol{b}^{Q_0}$. If the system is consistent (namely, (8) holds), then the solutions to (7) are exactly

$$\begin{cases} z_1 = e^{2k_1\pi/d_1}\zeta_1 & \text{for} \quad k_1 = 0, \ldots, d_1 - 1 \\ z_2 = e^{2k_2\pi/d_2}\zeta_2 & \text{for} \quad k_2 = 0, \ldots, d_2 - 1 \\ \qquad\vdots & \\ z_r = e^{2k_r\pi/d_r}\zeta_r & \text{for} \quad k_r = 0, \ldots, d_r - 1 \end{cases} \tag{10}$$

where each ζ_j is a fixed choice of the d_j-th root of the j-th coordinate of $\boldsymbol{b}^{Q}$. Clearly, all of them are isolated and the total number of these solutions is precisely $\prod_{j=1}^{r}|d_j| = |\det D|$. If $r < n$, then the $\mathbb{C}^*$-solution set of the decomposed system (7)–(9) breaks into "components" of the form $\{(e^{2k_1\pi/d_1}\zeta_1, \ldots, e^{2k_r\pi/d_r}\zeta_r, z_{r+1}, \ldots, z_n) : (z_{r+1}, \ldots, z_n) \in (\mathbb{C}^*)^{n-r}\}$, and they are in one-to-one correspondence with solutions in (10). Since each component is parametrized by the $n-r$ free variables $z_{r+1}, \ldots, z_n$, it is smooth and of dimension $n-r$.

To translate the above description of the $(\mathbb{C}^*)^n$-solution set of the decomposed system (in $\boldsymbol{z}$) into a description of the original solution set $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$, one may simply apply the change of variables $\boldsymbol{x} = \boldsymbol{z}^P$. Note that this map and its inverse $\boldsymbol{z} = \boldsymbol{x}^{P^{-1}}$ are both given by monomials (*bi-regular* maps (Hartshorne, 1977)), the basic properties of the solution set, such as, the number of solution components, their dimensions, and smoothness are therefore preserved. To summarize, under the assumption that the solution set is of positive dimension (that is $r < n$), the above elaborations assert the following proposition.

Proposition 1 *(Global parametrization (Eisenbud and Sturmfels, 1996; Kahle, 2010; Sturmfels, 1997)). For the solution set $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$ in $(\mathbb{C}^*)^n$, let P, Q, Q_0 and D be those matrices that appeared in the decompositions of A in (5) and (6). Assume $r := \operatorname{rank} A < n$.*

If $r < m$ and $\mathbf{1} \neq \boldsymbol{b}^{Q_0}$ then the binomial system is inconsistent, and therefore $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b}) = \varnothing$.

Otherwise (if $r = m$ or $\mathbf{1} = \boldsymbol{b}^{Q_0}$) $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$ consists of $\prod_{j=1}^{r}|d_j| = |\det D|$ components $V_{k_1,\ldots,k_r}$ for $k_1 \in \{0, \ldots, |d_1|-1\}, \ldots, k_r \in \{0, \ldots, |d_r|-1\}$. Each component $V_{k_1,\ldots,k_r}$ is smooth of dimension $n-r$, and it is parametrized by the smooth global parametrization $\phi_{k_1,\ldots,k_r} : (\mathbb{C}^*)^{(n-r)} \to V_{k_1,\ldots,k_r}$ given by

$$\phi_{k_1,\ldots,k_r}(t_1, \ldots, t_{n-r}) = (e^{2k_1\pi/d_1}\zeta_1, \ldots, e^{2k_r\pi/d_r}\zeta_r, t_1, \ldots, t_{n-r})^P \tag{11}$$

where each ζ_j is a fixed choice of the d_j-th root of the j-th coordinate of $\boldsymbol{b}^Q$.

Here we focus on the positive dimensional case. That is, in the following we restrict our attention to cases where $n - r > 0$ and the system is consistent. In this situation, for both theoretical interests and demands from concrete applications, one often wishes to identify another important property: the *degrees* of each component in $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$. Degree is a classic concept developed for plane algebraic curves. For example, the quadratic equation $y - x^2 = 0$ defines a curve of degree 2, i.e., the parabola. The generalized notation of degree for irreducible algebraic sets is usually formulated algebraically via Hilbert Polynomials. In this article, following the common practice of Numerical Algebraic Geometry, we shall take a geometric approach: Let $V = V_{k_1,\ldots,k_r}$ be a component of $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$ for some fixed choice of $k_1, \ldots, k_n$ as defined in Proposition 1. The number of isolated intersection points between V and a "generic" affine space of complementary dimension is a fixed number, and this number is the **degree** of V, denoted by $\deg V$.

Stated more precisely, let $\mathcal{G}_r$ be the set of all affine space in $\mathbb{C}^n$ of dimension $r = n - \dim V$. Then it can be shown that in a fixed open and dense subset of $\mathcal{G}_r$, all the affine spaces intersect with V at a fixed number of isolated points. This geometric interpretation of degree is explained in Fulton (1998), Hartshorne (1977), Sommese and Wampler (2005).

From a computational standpoint, a generic affine space in $\mathcal{G}_r$ can be represented by the solution set of a system of $d := n - r$ linear equations with generic coefficients.[1] Therefore $\deg V$ is precisely the number of points $\boldsymbol{x} = (x_1, \ldots, x_n) \in V$ that satisfies the linear system

[1] The representation of affine spaces by linear systems of equations is clearly not a one-to-one correspondence. E.g. for a matrix L, $L\boldsymbol{x} + \boldsymbol{b} = \mathbf{0}$ and $GL\boldsymbol{x} + G\boldsymbol{b} = \mathbf{0}$ define the same affine space for any nonsingular square matrix G of the appropriate dimension. The refined parametrization of the affine spaces is given the *Grassmannian*. However, in the current context, the representation by linear systems is sufficient as the fixed Zariski open and dense subset of r-dimensional affine spaces from which the $\deg V$ can be defined also corresponds to a Zariski open and dense subset among the linear systems (in terms of their coefficients).

$$\begin{cases} c_{11}x_1 + c_{12}x_2 + \cdots + c_{1n}x_n = c_{10} \\ \qquad\qquad\qquad\qquad \vdots \\ c_{d1}x_1 + c_{d2}x_2 + \cdots + c_{dn}x_n = c_{d0} \end{cases} \tag{12}$$

where c_{ij}'s for $i = 1, \ldots, d$ and $j = 0, \ldots, n$ are generic complex numbers (that is, $(c_{ij})_{i=1,\ldots,d,\, j=0,\ldots,n}$ belongs to a Zariski open and dense subset of $\mathbb{C}^{d\times(n+1)}$). But recall that the set $V = V_{k_1,\ldots,k_r}$ is precisely the image of the injective map

$$\phi_{k_1,\ldots,k_r}(t_1, \ldots, t_d) \;=\; (e^{2k_1\pi/d_1}\zeta_1, \ldots, e^{2k_r\pi/d_r}\zeta_r, t_1, \ldots, t_d)^P$$

in Proposition 1. If we let $\xi = (e^{2k_1\pi/d_1}\zeta_1, \ldots, e^{2k_r\pi/d_r})$ and $\boldsymbol{t} = (t_1, \ldots, t_d)$ then

$$\phi_{k_1,\ldots,k_r}(\boldsymbol{t}) = (\xi, \boldsymbol{t})^{\binom{P_r}{P_0}} = (\xi^{\boldsymbol{p}_r^{(1)}}\boldsymbol{t}^{\boldsymbol{p}_0^{(1)}}, \;\ldots,\; \xi^{\boldsymbol{p}_r^{(n)}}\boldsymbol{t}^{\boldsymbol{p}_0^{(n)}})$$

where for each $j = 1, \ldots, n$, $\boldsymbol{p}_r^{(j)}$ and $\boldsymbol{p}_0^{(j)}$ are the j-th columns of P_r and P_0 respectively. In other words, V has the global parametrization $x_i = \xi^{\boldsymbol{p}_r^{(1)}}\boldsymbol{t}^{\boldsymbol{p}_0^{(1)}}$. Therefore the intersections between V and the generic affine space defined by (12) are precisely the solutions of the polynomial system

$$\begin{cases} c_{11}\,\xi^{\boldsymbol{p}_r^{(1)}}\boldsymbol{t}^{\boldsymbol{p}_0^{(1)}} + c_{12}\,\xi^{\boldsymbol{p}_r^{(2)}}\boldsymbol{t}^{\boldsymbol{p}_0^{(2)}} + \cdots + c_{1n}\,\xi^{\boldsymbol{p}_r^{(n)}}\boldsymbol{t}^{\boldsymbol{p}_0^{(n)}} = c_{10} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ c_{d1}\,\xi^{\boldsymbol{p}_r^{(1)}}\boldsymbol{t}^{\boldsymbol{p}_0^{(1)}} + c_{d2}\,\xi^{\boldsymbol{p}_r^{(2)}}\boldsymbol{t}^{\boldsymbol{p}_0^{(2)}} + \cdots + c_{dn}\,\xi^{\boldsymbol{p}_r^{(n)}}\boldsymbol{t}^{\boldsymbol{p}_0^{(n)}} = c_{d0} \end{cases}$$

By letting $c'_{ij} := c_{ij}\xi^{\boldsymbol{p}_r^{(j)}}$ and $c'_{i0} := c_{i0}$ for each $i = 1, \ldots, d$ and $j = 1, \ldots, n$, the above is a system of d Laurent polynomial equations in variables $\boldsymbol{t} = (t_1, \ldots, t_d)$ with coefficients c'_{ij} and the same set of monomials $\boldsymbol{t}^{\boldsymbol{p}_0^{(j)}}, \ldots, \boldsymbol{t}^{\boldsymbol{p}_0^{(j)}}$.

Note that the transformation from (c_{ij}) to (c'_{ij}), given by $c'_{ij} := c_{ij}\xi^{\boldsymbol{p}_r^{(j)}} \in \mathbb{C}$ for $j \neq 0$, is a non-singular linear transformation which can be represented by a diagonal matrix with nonzero entries $\xi^{\boldsymbol{p}_r^{(j)}}$ (since $\xi \in (\mathbb{C}^*)^r$). This linear transformation preserves the Zariski topology. In other words, the "generic" choices of (c_{ij}) translate to "generic choices" of (c'_{ij}). We summarize the above elaboration into the following proposition.

Proposition 2 *(Degree via affine space cut). Assuming $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$ is d-dimensional, then the degree of a component V of $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$ is the number of solutions $\boldsymbol{t} \in (\mathbb{C}^*)^d$ of the system of d Laurent polynomial equations*

$$\begin{cases} c_{11}\boldsymbol{t}^{\boldsymbol{p}_0^{(1)}} + c_{12}\boldsymbol{t}^{\boldsymbol{p}_0^{(2)}} + \cdots + c_{1n}\boldsymbol{t}^{\boldsymbol{p}_0^{(n)}} = c_{10} \\ \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ c_{d1}\boldsymbol{t}^{\boldsymbol{p}_0^{(1)}} + c_{d2}\boldsymbol{t}^{\boldsymbol{p}_0^{(2)}} + \cdots + c_{dn}\boldsymbol{t}^{\boldsymbol{p}_0^{(n)}} = c_{d0} \end{cases} \tag{13}$$

for coefficients (c_{ij}) in a fixed Zariski open and dense subset of $(\mathbb{C}^*)^{d\times(n+1)}$.

It is important to note that for generic coefficients, the $\mathbb{C}^*$-solutions of the above system are all isolated (0-dimensional), and the total number is a constant. By the Kushnirenko's Theorem (Kushnirenko, 1975) (or the more general Bernshtein's Theorem (Bernshtein, 1975; Huber and Sturmfels, 1995)), this number can be expressed in terms of the volume of the *Newton polytope* of the above

system which is the convex hull of the set of n points $\boldsymbol{p}^{(1)}, \ldots \boldsymbol{p}^{(n)}, \mathbf{0}$ in $\mathbb{R}^d$.[2] Here we state the result in the context of degree computation and leave the technical statement of the theorem, as well as its related concepts to Appendix A.

Proposition 3 *(Degree as volume). The generic number of isolated solutions $\boldsymbol{t} \in (\mathbb{C}^*)^d$ of the system of Laurent polynomial equation (13) and hence the degree of V is*

$$\deg V = d\,! \cdot \mathrm{Vol}_d(\mathrm{conv}\{\boldsymbol{p}_0^{(1)}, \ldots, \boldsymbol{p}_0^{(n)}, \mathbf{0}\}) \tag{14}$$

where $\mathbf{0} = (0, \ldots, 0)^\top \in \mathbb{R}^d$ and columns $\boldsymbol{p}_0^{(1)}, \ldots, \boldsymbol{p}_0^{(n)}$ of the matrix P_0 are considered as points in $\mathbb{R}^d$. The notation conv *denotes the operation of taking convex hull, and* Vol_d *is the volume of a convex body in $\mathbb{R}^d$.*

With this formulation, the degree of a component of $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$ can be computed efficiently through methods in combinatorial geometry.

Remark 4. Recall that while the Smith Normal Form $\left[\begin{smallmatrix} D & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right]$ of A is unique up to a change of signs of the diagonal entries in D, the transformation matrices P and Q are generally not unique. However, different choices of the transformations will produce the same value for $\deg V$ as given in (14). This invariance is best understood via the notion of "local degree" (Gelfand et al., 1994) of a component V of $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$ which is a generalization of the concept of *multiplicity* and naturally independent from the transformation matrices P and Q used in the Smith Normal Form computation. This approach, however, requires more advanced mathematical tools. We therefore provide a simple geometric justification:

Suppose there is a different pair of unimodular matrices $\tilde{P} = \left[\begin{smallmatrix} \tilde{P}_r \\ \tilde{P}_0 \end{smallmatrix}\right] \in M_{n\times n}(\mathbb{Z})$ where $\tilde{P}_0 \in M_{(n-r)\times n}(\mathbb{Z})$ and $\tilde{Q} \in M_{m\times m}(\mathbb{Z})$ such that

$$\begin{bmatrix} \tilde{P}_r \\ \tilde{P}_0 \end{bmatrix} A \begin{bmatrix} \tilde{Q}_r & \tilde{Q}_0 \end{bmatrix} = \begin{bmatrix} D & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Then it can be shown that there is a unimodular matrix $G \in M_{(n-r)\times(n-r)}(\mathbb{Z})$ such that

$$\tilde{P}_0 = G\, P_0.$$

The simple proof is included in the Appendix B. Since G is unimodular, that is $|\det G| = 1$, it does not change the volume of the "parallelepiped" spanned by the columns of P_0 (as vectors). In other words, letting $\tilde{\boldsymbol{p}}_0^{(1)}, \ldots, \tilde{\boldsymbol{p}}_0^{(n)}$ be the columns of $\tilde{P}_0$, then

$$\mathrm{Vol}_d(\mathrm{conv}\{\boldsymbol{p}_0^{(1)}, \ldots, \boldsymbol{p}_0^{(n)}, \mathbf{0}\}) = \mathrm{Vol}_d(\mathrm{conv}\{\tilde{\boldsymbol{p}}_0^{(1)}, \ldots, \tilde{\boldsymbol{p}}_0^{(n)}, \mathbf{0}\}).$$

Therefore, the volume expression used in Proposition 3, despite the apparent dependency, is actually independent from the choices of transformation matrices used in (5).

3. Parallel degree computation

Proposition 3 provides a computationally viable means for computing the degree of each component as the normalized volume of a convex polytope. In this section we shall present a parallel algorithm for computing the degree that is suitable for both multi-core systems and GPUs, though the focus is the GPU-based implementation. In this section we focus on the computation of (14). Let $S = \{\boldsymbol{p}_0^{(1)}, \ldots, \boldsymbol{p}_0^{(n)}\} \subset \mathbb{R}^d$, then

[2] In Gelfand et al. (1994) the degree is shown to be equivalent to the "local degree" at a certain point which, in turn, is proved to be the volume of the Newton polytope. The Kushnirenko's Theorem is then derived as a corollary of this fact. However, the original proof of Bernshtein's Theorem and the alternative proof provided by Huber and Sturmfels (1995) do not make direct use of the expression of the degree of the $\mathbb{C}^*$-solution set of a binomial system.

$$\deg V = d!\,\mathrm{Vol}_d(\mathrm{conv}\, S). \tag{15}$$

For brevity, let $\mathrm{NVol}_d = d!\,\mathrm{Vol}_d$ be the **normalized volume** in $\mathbb{R}^d$, then the above equation becomes

$$\deg V = \mathrm{NVol}_d(\mathrm{conv}\, S) \tag{16}$$

Therefore the problem of finding $\deg V$ is equivalent to the computation of the normalized volume of a *lattice polytope* (a polytope whose vertices have integer coordinates).

Remark 5. Clearly, (15) and (16) are equivalent. However, from the computational point of view, there is one crucial distinction: The knowledge that $\mathrm{NVol}_d(S)$ must be an integer permits the use of efficient but potentially less accurate numerical methods using floating point arithmetic and still obtain the correct result. Indeed, the exact results can still be obtained as long as the total absolute error is kept below 1/2. This is not possible for methods that are designed to compute volume of more general polytopes. The algorithm, for computing (16), to be presented below, is hence not directly comparable to *exact volume computation* algorithms (Bárány and Füredi, 1987; Büeler et al., 2000) for general polytopes.

3.1. Regular simplicial subdivision

Among many different approaches for computing the normalized volume, here we adopt a classic technique known as *regular simplicial subdivision* (Loera et al., 2010) as this approach produces an important byproduct that will be used in the computation of witness set, which will be the subject of §4. In this approach, we are interested in computing the normalized volume $\mathrm{NVol}_d(\mathrm{conv}\, S)$ by dividing the lattice polytope $\mathrm{conv}\, S$ into a collection of smaller pieces for which the volume computation is easy. Without loss of generality, we assume S has no interior points, that is, no point in S lies in the interior of $\mathrm{conv}\, S$.

Definition 6. A **cell** of S is simply an affinely independent subset of S. A **simplicial subdivision** of S is a collection $\mathcal{D}$ of cells of S, such that

1. For each $C \in \mathcal{D}$, $\mathrm{conv}\, C$ is a d-simplex inside $\mathrm{conv}\, S$;
2. For any distinct pair of simplices $C_1, C_2 \in \mathcal{D}$, the intersection of $\mathrm{conv}\, C_1$ and $\mathrm{conv}\, C_2$, if nonempty, is a common face of the two; and
3. The union of convex hulls of all cells in $\mathcal{D}$ is exactly $\mathrm{conv}\, S$.

A simplicial subdivision plays an important role in computing $\mathrm{NVol}_d(\mathrm{conv}\, S)$: the normalized volume of a d-simplex in $\mathbb{R}^d$ is easy to compute: given a d-simplex $\Delta = \mathrm{conv}\{\boldsymbol{a}_0, \ldots, \boldsymbol{a}_d\} \subset \mathbb{Z}^d$,

$$\mathrm{NVol}_d(\Delta) = |\det\begin{bmatrix}\boldsymbol{a}_1 - \boldsymbol{a}_0 & \cdots & \boldsymbol{a}_d - \boldsymbol{a}_0\end{bmatrix}|. \tag{17}$$

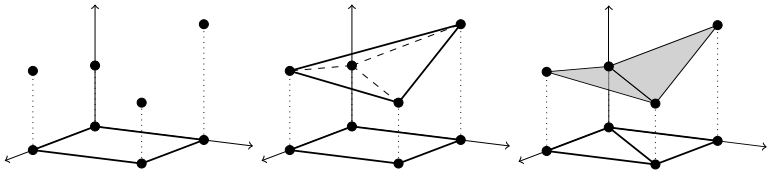
So the volume of $\mathrm{conv}\, S$ can be computed easily as the sum of the volume of all simplices in $\mathcal{D}$.

Note that the simplicial subdivision for a given polytope is, in general, not unique, and there are many different approaches for constructing them. Here we focus on the approach of regular simplicial subdivision: One can define a "lifting function" $\omega : S \to \mathbb{R}$ by assigning a real number to each point in S. For each point $\boldsymbol{a} \in S$, a new point $(\boldsymbol{a}, \omega(\boldsymbol{a})) \in \mathbb{R}^n$ can be created by using $\omega(\boldsymbol{a})$ as an additional coordinate. This procedure "lifts" points of S into $\mathbb{R}^{d+1}$, the space of one higher dimension. Let

$$\hat{S} = \{\hat{\boldsymbol{a}} = (\boldsymbol{a}, \omega(\boldsymbol{a})) \mid \boldsymbol{a} \in S\} \tag{18}$$

be the lifted version of S via the lifting function ω. Figs. 1a and 1b show examples of this lifting procedure. Let $\pi : \mathbb{R}^{d+1} \to \mathbb{R}^d$ be the projection that simply erases the last coordinate, then $\pi(\hat{S}) = S$.

Recall that for a face $\hat{F}$ of the lifted polytope $\mathrm{conv}\,\hat{S}$, its *inner normal* is a vector $\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{d+1}$ such that the linear functional $\langle \bullet, \hat{\boldsymbol{\alpha}} \rangle$ attains its minimum over $\mathrm{conv}\,\hat{S}$ on $\hat{F}$. Moreover, a face $\hat{F}$ of $\mathrm{conv}\,\hat{S}$ is called a **lower face** with respect to the projection π if its inner normal $\hat{\boldsymbol{\alpha}}$ has positive last coordinate. Without loss of generality, in this case, we may assume the last coordinate of $\hat{\boldsymbol{\alpha}}$ to be 1, that is,



(a) Lifted set $\hat{S}$ (b) Lifted polytope conv $\hat{S}$ (c) Projection of lower hull

Fig. 1. Regular simplicial subdivision via generic lifting.

$\hat{\boldsymbol{\alpha}} = (\alpha_1, \ldots, \alpha_n, 1) \in \mathbb{R}^{d+1}$. It can be shown that if the lifting function $\omega : S \to \mathbb{R}$ is chosen so that no $j+2$ points in $\hat{S}$ are contained in a j-dimensional affine space, then the projections of all the d-dimensional lower faces of conv $\hat{S}$ via π form a simplicial subdivision for conv S which is called a *regular simplicial subdivision* of conv S. Such a lifting function is called *generic* (with respect to S). In particular, if the liftings are randomly picked, then with probability one, the lifting function will be generic. The construction of this simplicial subdivision is therefore equivalent to the enumeration of all the lower faces of conv $\hat{S}$.

Example 7. Consider, for example, $S = \{(0,0), (0,1), (1,1), (1,0)\}$ in the xy-plane. A simplicial subdivision of conv S can be obtained via the following procedure: First assign "liftings" $\omega_1, \omega_2, \omega_3, \omega_4 \in \mathbb{R}$ to each of the vertices as the z-coordinate and obtain new points $(0,0,\omega_1)$, $(0,1,\omega_2)$, $(1,1,\omega_3)$, $(1,0,\omega_4)$ in $\mathbb{R}^3$. It is easy to verify that with almost all choices of the liftings the four "lifted" points (Fig. 1a) do not lie on the same plane. In that case, the convex hull conv$\{(0,0,\omega_1), (0,1,\omega_2), (1,1,\omega_3), (1,0,\omega_4)\}$ of the four lifted vertices form a three dimensional polytope (Fig. 1b) with triangle faces. Of particular importance is the lower hull of this polytope which is the faces facing downward. As shown in Fig. 1c, the projection of the faces in the lower hull back onto the xy-plane forms a simplicial subdivision of the original shape conv S.

Algebraically speaking, a d-dimensional lower face of conv S is the convex hull of a set of $d+1$ points $\{\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_d\} \subset \hat{S}$ for which there exists a $\hat{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}, 1) \in \mathbb{R}^{d+1}$ such that the system of inequalities

$$I(\boldsymbol{a}_0, \ldots, \boldsymbol{a}_d) \; : \; \begin{cases} \langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle = \langle \hat{\boldsymbol{a}}_j, \hat{\boldsymbol{\alpha}} \rangle & \text{for } j = 1, \ldots, d \\ \langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle \le \langle \hat{\boldsymbol{a}}, \hat{\boldsymbol{\alpha}} \rangle & \text{for } \boldsymbol{a} \in S \end{cases} \tag{19}$$

is satisfied. In other words, the existence of the lower face defined by $\{\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_d\} \subset \hat{S}$ is equivalent to the feasibility of the above system of inequalities $I(\boldsymbol{a}_0, \ldots, \boldsymbol{a}_d)$. This algebraic description of the lower faces is the basis on which enumeration methods are developed. In the following subsections, we shall present an approach that results in a *parallel algorithm* which is suitable for both multi-core systems and GPU devices. In this approach, we employ two complementary processes of "extension" and "pivoting". We shall outline them below.

3.2. Extension of k-faces

Intuitively speaking, in the extension process, one starts with the vertices of the lower hull of conv $\hat{S}$. For each of these vertices, systematic attempts are made to "extend" it by finding another lower vertex so that the two vertices form a "lower edge" (an edge on the lower hull of conv $\hat{S}$). The possible extensions may not be unique, and for each possibility, further attempts are made to extend it to 2-dimensional lower faces. This process continues until one reaches all the d-dimensional lower faces. Finally, the collection of such d-dimensional lower faces will project down, via π, to form a simplicial subdivision for conv S.

To describe this process, we first extend the characterization (19) to include lower faces of all dimensions: A set of affinely independent $k+1$ points in $\hat{S}$ is said to determine a **lower k-face** if their convex hull forms a k-dimensional lower face of conv $\hat{S}$ with respect to the projection π. Stated

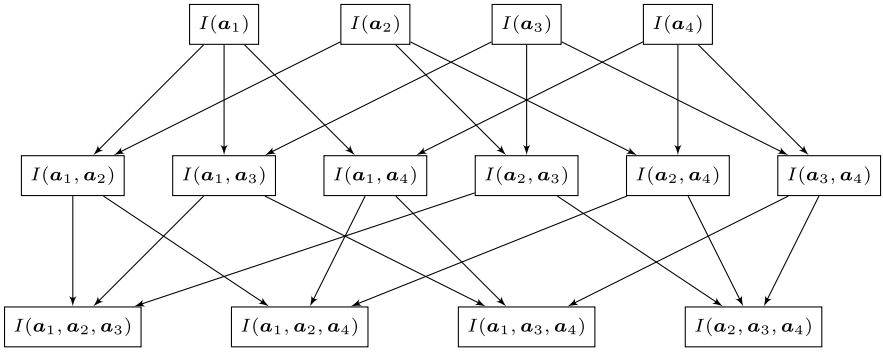


Fig. 2. A directed acyclic graph of possible lower k-faces.

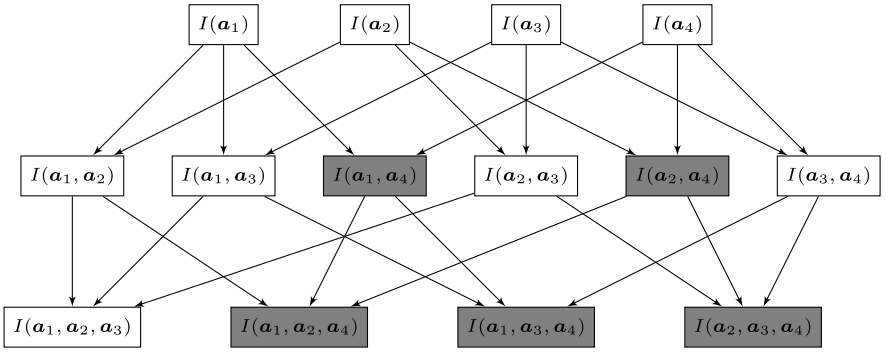


Fig. 3. A direct graph of possible lower k-faces colored by the feasibility of the corresponding system of inequalities.

algebraically, the affinely independent set $\{\boldsymbol{a}_0, \ldots, \boldsymbol{a}_k\}$ determines a lower k-face if and only if there exists an $\hat{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}, 1) \in \mathbb{R}^{d+1}$, such that the system of inequalities

$$I(\boldsymbol{a}_0, \ldots, \boldsymbol{a}_k) \; : \; \begin{cases} \langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle = \langle \hat{\boldsymbol{a}}_j, \hat{\boldsymbol{\alpha}} \rangle & \text{for } j = 1, \ldots, k \\ \langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle \leq \langle \hat{\boldsymbol{a}}, \hat{\boldsymbol{\alpha}} \rangle & \text{for } \boldsymbol{a} \in S \end{cases} \tag{20}$$

is satisfied.

Geometrically, a lower 0-face is a vertex on the lower hull of $\operatorname{conv} \hat{S}$, and a lower 1-face is simply a lower edge, etc. We can conveniently organize all possible system of inequalities of the above form into a *directed acyclic graph*, as illustrated in Fig. 2, where each node represents a system of inequalities and there is an edge from $I(A)$ to $I(B)$ whenever B is obtained by joining new points in S into A. With this construction, the resulting graph is *graded* by the number of points involved.

It can be easily verified that for generic lifting function ω, containment relation between lower k-faces of the same dimension is impossible. That is, for a fixed k, no lower k-face is contained in another lower k-face. Therefore the graph describes precisely the containment relationship among possible lower k-faces.

A node is said to be feasible if the corresponding system of inequality is feasible. Fig. 3 shows an example of the labeling of the graph via the feasibility of the nodes: dark for infeasible nodes and white for feasible ones. Recall that a node determines a lower k-faces if and only if it is feasible. Hence we only need to explore the feasible subgraph (the white subgraph in Fig. 3).

One crucial observation is that if two points do not define a lower edge, then they cannot be a part of any lower faces. More generally, if a set of points does not define a lower k-face, then there are no lower j-faces containing them for any $j > k$. Stated formally, for $\hat{F_1} \subset \hat{S}$,

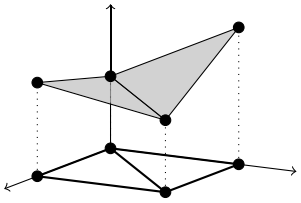


Fig. 4. Simplicial pivoting procedure moves from one lower face to another.

$$I(\hat{F}_1) \text{ is infeasible} \quad \Longrightarrow \quad I(\hat{F}) \text{ is infeasible for all } \hat{F}_1 \subset \hat{F} \subset \hat{S}.$$

In terms of the graph, if a node is infeasible, then the entire subgraph reachable by that node is infeasible. Therefore during the exploration of the graph, once an infeasible node is encountered, no further exploration from that node is needed as all nodes reachable are infeasible. This simple observation produces significant savings in terms of computation.

A key procedure in the exploration of the feasible subgraph is the jump from one feasible node to another along an edge. Assuming, for some $\{\boldsymbol{a}_0, \ldots, \boldsymbol{a}_k\} \subset S$, the node $I(\boldsymbol{a}_0, \ldots, \boldsymbol{a}_k)$ is feasible, then the feasibility of an adjacent node, say via the edge corresponds to $\boldsymbol{a}_{k+1}$, can be determined by solving the *linear programming problem* $LP(\boldsymbol{a}_0, \ldots, \boldsymbol{a}_k\,;\, \boldsymbol{a}_{k+1})$ given by

$$\begin{aligned} &\text{Minimize} \quad \langle \hat{\boldsymbol{a}}_{k+1}, \hat{\boldsymbol{\alpha}} \rangle - \langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle \\ &\text{subject to} \begin{cases} \langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle = \langle \hat{\boldsymbol{a}}_j, \hat{\boldsymbol{\alpha}} \rangle & \text{for } j = 1, \ldots, k \\ \langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle \le \langle \hat{\boldsymbol{a}}, \hat{\boldsymbol{\alpha}} \rangle & \text{for all } \boldsymbol{a} \in S \end{cases} \end{aligned} \tag{21}$$

in the variable $\hat{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}, 1)$ for $\boldsymbol{\alpha} \in \mathbb{R}^d$.

Note that under the constraints, the value of the objective function must be nonnegative. Indeed, the minimum value of 0 is attainable precisely when there is an $\hat{\boldsymbol{\alpha}}$ for which the constraints are satisfied and simultaneously $\langle \hat{\boldsymbol{a}}_{k+1}, \hat{\boldsymbol{\alpha}} \rangle = \langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle$. That is, minimum value is 0 if and only if $I(\boldsymbol{a}_0, \ldots, \boldsymbol{a}_k, \boldsymbol{a}_{k+1})$ is feasible. In this case, the new feasible node $I(\boldsymbol{a}_0, \ldots, \boldsymbol{a}_k, \boldsymbol{a}_{k+1})$ is discovered. Geometrically, we have "extended" the lower k-face determined by $\{\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_k\}$ into a lower $(k+1)$-face by joining it the new vertex $\hat{\boldsymbol{a}}_{k+1}$.

Using the extension procedure as a basic building block, one can gradually explore the feasible subgraph in an inherently parallel manner: different branches of the spanning tree of the subgraph can be explored independently.

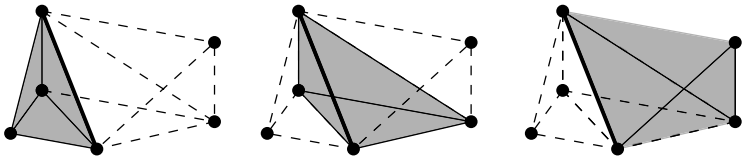
3.3. Simplicial pivoting

The extension procedure described above can be complemented by another process, called "simplicial pivoting", that explores the feasible subgraph by "moving sideways" in the graph from one lower d-face to another.

This process starts with a lower d-face of $\operatorname{conv} \hat{S}$ already obtained (which corresponds to a cell in the regular simplicial subdivision). Consider, for example, one of the lower faces shown in Fig. 4. Using an edge as a hinge, we shall "pivot" one lower face until another lower face is obtained. More generally, recall that a lower d-face is determined by a set of $d+1$ affinely independent points $\{\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_d\}$ in $\hat{S}$ that has an inner normal of the form $\hat{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}, 1)$ with $\boldsymbol{\alpha} \in \mathbb{R}^{d+1}$. Stated algebraically, the system of inequalities

$$\begin{aligned} &\langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle = \langle \hat{\boldsymbol{a}}_j, \hat{\boldsymbol{\alpha}} \rangle \quad \text{for } j = 1, \ldots, d \\ &\langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle \le \langle \hat{\boldsymbol{a}}, \hat{\boldsymbol{\alpha}} \rangle \quad \text{for all } \boldsymbol{a} \in S \end{aligned} \tag{22}$$

is satisfied. Note that the d equalities form a system of d linearly independent constrains on $\boldsymbol{\alpha} \in \mathbb{R}^d$ and hence uniquely determine $\boldsymbol{\alpha}$. By removing a single equality from the above system, we give the



(a) A simplex (b) A simplex obtained by a single pivoting operation (c) Another simplex obtained through further pivoting operation

Fig. 5. Via the pivoting procedure, one moves from a simplex (a) to a different simplex (b) by leaving a chosen vertex and then to yet another simplex (c).

inner normal $\hat{\boldsymbol{\alpha}}$ one degree of freedom which would allow it to "pivot". The goal is to let it pivot until it defines a different lower d-face.

For any choice $i = 0, \ldots, d$, with the equality corresponding to $\hat{\boldsymbol{a}}_i$ in the above system (22) removed, the inner normal $\hat{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}, 1) \in \mathbb{R}^d$, now with one degree of freedom, is characterized by the system

$$P(\boldsymbol{a}_0, \ldots, \boldsymbol{a}_d ; i) \; : \; \begin{cases} \langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle = \langle \hat{\boldsymbol{a}}_j, \hat{\boldsymbol{\alpha}} \rangle & \text{for } j = 1, \ldots, d, \text{ but } j \neq i \\ \langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle \lesssim \langle \hat{\boldsymbol{a}}_i, \hat{\boldsymbol{\alpha}} \rangle & \\ \langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle \leq \langle \hat{\boldsymbol{a}}, \hat{\boldsymbol{\alpha}} \rangle & \text{for all } \boldsymbol{a} \in S \end{cases} \tag{23}$$

Note that this system has $d - 1$ equalities. If a solution with d equalities exists, then that solution corresponds to a different lower d-face (i.e. a different cell in the resulting subdivision). In the context of Linear Programming, such a solution is called a *basic feasible solution*. The problem of finding a basic feasible solution is known as the *Phase One* problem in Linear Programming. It can be solved accurately and efficiently.

This procedure is called "simplicial pivoting". It allows us to pivot from one lower d-face to another. By repeatedly applying this procedure, more lower d-faces can be gathered. Fig. 5 illustrates this process.

3.4. Traverse the feasible subgraph

In the above we have formulated the enumeration of lower d-faces as the problem of exploring the feasible subgraph which contains them as a subset. We also have two procedures for "walking" within the graph: The extension procedure moves from one lower face to another of one higher dimension while the pivoting procedure jumps from one lower d-face to another lower d-face. With these building blocks in place, the exploration can be handled by a classic *graph traversal algorithm* following a "discover-and-explore" scheme. This algorithm can be implemented to exploit the inherent parallelism: the neighbors of each discovered node can be explored independently. The detailed descriptions of such an algorithm can be found in standard text in parallel algorithms (e.g. Skiena, 2009; Herlihy and Shavit, 2012). Of particular relevance is the adoption of such graph traversal algorithm in the context of "mixed cell enumeration". The connection is discussed in §5. Here we focus on the issues specific to an efficient GPU-based implementation.

One important problem we must deal with, in the parallel algorithm, is that same nodes may be discovered by different threads at the same time. Since the degree is the sum of the normalized volume of the projection of all the lower d-faces which are represented by the nodes in the feasible subgraph, duplicated nodes will produce incorrect results. Therefore, an efficient and scalable mechanism for ensuring no duplicated lower d-faces are listed is the key to the correctness of the algorithm.

This mechanism appears to be the bottleneck, in terms of performance, in related algorithms with similar organization (see §5). Our experiments confirm that an inefficient checking mechanism would be the limiting factor of the scalability in a parallel implementation. Since on a GPU, it is typical to have thousands of threads active simultaneously, the efficiency of such a mechanism is crucial.

In the present contribution, the *hash table* data structure is used to keep track of the nodes, in the graph, that have been discovered or completely-explored. The great advantage of this choice is that unlike a sorted data structure, hash table provides nearly constant access time, in *most* cases. In our current implementation, for simplicity, the well-tested bit-string hash function from the standard C++ library is used.[3] Our experiments suggest that a hash table with 2^{16} to 2^{20} entries[4] is sufficient for all problems considered in our numerical experiments to be presented in §6 in the sense that the collision rate within hash table access can be virtually ignored.

3.5. Summary of the algorithm

In the above, we formulate the degree computation for solution components defined by binomial systems as the exploration of the feasible subgraph to be accomplished by the two complementary processes: extension and pivoting. In this section, we list the main algorithms.

These algorithms are designed for a system with one or more GPU devices and a single CPU with the GPU performing most of the computationally intensive tasks. For simplicity, we restrict ourselves to modern GPUs manufactured by NVidia and build our program based on NVidia CUDA (a GPU programming framework). The GPU devices must share memory with the CPU since they must all have access to data structures `WaitingNodes`, `KnownNodes`, and `NewNodes` (to be defined below). In the current implementation, this is accomplished via a technique known as *pinned memory* (NVIDIA Corporation, 2011) provided by the CUDA framework which allows the efficient sharing of data between GPU devices and the CPU without explicit copying.

In the following algorithms, the list `WaitingNodes` contains nodes whose feasibility is to be determined by the extension procedure. `Cells` is the unordered collection of lower d-faces already discovered. `KnownNodes` is the hash table that records the discovery of nodes, and it is crucial mechanism by which we ensure the uniqueness of the discovered nodes. Finally, `NewNodes` is an unordered list that keeps track of nodes discovered through pivoting or extension. They need to be checked against `KnownNodes` for uniqueness.

RANDOM is a function that randomly chooses an item from a collection using pseudo random number generator. The randomness is employed to achieve a more uniform performance from one run to another for the sole purpose of simplifying the benchmarking process. SIMPLEXPHASEONE and SIMPLEXPHASETWO are the phase-one and phase-two algorithms of the simplex method for the linear programming problems (21) and (23) respectively. Even though, at over 3000 lines, the C++ code for these two components are the longest and most complicated parts of the entire program, they have been a fixture of the long line of "mixed volume computation" software developed over the last two decades whence the present work inherits much of its techniques and design. Therefore we choose to not describe them in detail and refer to works such as Chen et al. (2014b), Gao and Li (2000, 2003), Lee and Li (2011), Li and Li (2001), Mizutani and Takeda (2008), Mizutani et al. (2007).

The EXTEND procedure tests the feasibility of a node (see §3.2) in the waiting list `WaitingNodes`, and it is designed to run simultaneously on all available threads across all GPU devices.

```
1: function EXTEND
2:     if WaitingQueue ≠ ∅ then
3:         {a_0, ..., a_k} ← DEQUEUE( WaitingQueue )
4:         F ← SIMPLEXPHASETWO( LP({a_0, ..., a_k}) )
5:         if F ≠ ∅ then
6:             NewNodes ← NewNodes ∪ {F}
7:         end if
8:     end if
9: end function
```

[3] The specialized hash function for bit-strings (std::bitset) is a new addition to C++11 standard. In our implementation, the version as in LLVM (The LLVM Compiler Infrastructure Project) was used.

[4] Each entry is a 32-bit integer that represents the index of a known node which, in turn, corresponds to a cell in the regular simplicial subdivision of conv S.

The PIVOT procedure implements the simplicial pivoting process detailed in §3.3, and it is designed to run simultaneously on all available threads across all GPU devices. It picks a random lower d-face already discovered and applies simplicial pivoting to potentially obtain a new lower faces. Just like the EXTEND procedure above, newly discovered nodes will be placed in the `NewNodes` list.

```
1: function PIVOT
2:     if Cells ≠ ∅ then
3:         {a_0, ..., a_d} ← RANDOM( Cells )
4:         ℓ ← min(d + 1, 10)
5:         for i = 1, ..., ℓ do
6:             j ← RANDOM( {0, ..., d} )
7:             F ← SIMPLEXPHASEONE( P({a_0, ..., a_d} \ {a_j}) )
8:             if F ≠ ∅ then
9:                 NewNodes ← NewNodes ∪ {F}
10:            end if
11:        end for
12:    end if
13: end function
```

The procedure CHECKUNIQ checks newly discovered nodes against the hash table `KnownNodes` to make sure they have not already been discovered. It will run on a GPU device with a large number of threads simultaneously checking the uniqueness of all nodes in the list of `NewNodes`.

```
1: function CHECKUNIQ
2:     if NewNodes ≠ ∅ then
3:         {a_0, ..., a_k} ← DEQUEUE( NewNodes )
4:         if {a_0, ..., a_k} ∉ KnownNodes then
5:             KnownNodes = KnownNodes ∪ {F}
6:             if k = d + 1 then
7:                 Cells = Cells ∪ {{a_0, ..., a_k}}
8:             else
9:                 for all a ∈ S \ {a_0, ..., a_k} do
10:                    if {a_0, ..., a_k, a} ∉ KnownNodes then
11:                        WaitingQueue = WaitingQueue ∪ {{a_0, ..., a_k, a}}
12:                    end if
13:                end for
14:            end if
15:        end if
16:    end if
17: end function
```

Finally, the main procedure, which runs on the CPU, coordinates all the different processes.

```
1: function MAIN
2:     WaitingNodes ← S
3:     while WaitingNodes ≠ ∅ do
4:         Run EXTEND on available GPU threads
5:         Run PIVOT on available GPU threads
6:         Wait for EXTEND and PIVOT
7:         Run CHECKUNIQ on available GPU threads
8:     end while
9: end function
```

Note that the EXTEND and PIVOT are two independent procedures (i.e. two different "kernels" in the CUDA terminology) that can be executed in parallel. In particular, when more than one GPU devices are available in a system, the two procedures can run on different devices to maximize the scalability.

4. Computation of witness sets

The concept of "witness sets" (Sommese and Wampler, 1996) is one of the most fundamental and versatile tools in Numerical Algebraic Geometry. In its most basic form, given a pure dimensional algebraic set, it can be shown that its intersection with a generic affine space of complementary dimension consists of finitely many isolated points. This finite set is called a *witness set* of the algebraic set. It is a key ingredient in the computation of Numerical Irreducible Decomposition (Sommese and Wampler, 1996) of an algebraic set. In many scenarios, it produces the degree of each component as a byproduct. Indeed, this technique (via witness sets) was first used to numerically compute the degrees of the "Master Space" problem (§6) in the work by Hauenstein et al. (2013).

Given the ubiquity of witness sets in Numerical Algebraic Geometry, in this section, we shall briefly outline a homotopy construction for computing witness sets for a component of $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$ that utilizes the extra information generated by the degree computation process described above. It is a special case of the *polyhedral homotopy* (Huber and Sturmfels, 1995).

Recall that by Proposition 2, the intersection between a component $V \subseteq \mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$ and a generic affine space of complementary dimension consists of precisely the points $\boldsymbol{t} = (t_1, \ldots, t_d) \in (\mathbb{C}^*)^d$ that satisfy the system of d Laurent polynomial equations in d variables given by

$$\begin{aligned} c_{11}\boldsymbol{t}^{\boldsymbol{p}_0^{(1)}} + c_{12}\boldsymbol{t}^{\boldsymbol{p}_0^{(2)}} + \cdots + c_{1n}\boldsymbol{t}^{\boldsymbol{p}_0^{(n)}} &= c_{10} \\ &\vdots \\ c_{d1}\boldsymbol{t}^{\boldsymbol{p}_0^{(1)}} + c_{d2}\boldsymbol{t}^{\boldsymbol{p}_0^{(2)}} + \cdots + c_{dn}\boldsymbol{t}^{\boldsymbol{p}_0^{(n)}} &= c_{d0} \end{aligned} \tag{24}$$

where the coefficients depend on both the choice of the component in $\mathcal{V}^*(\boldsymbol{x}^A - \boldsymbol{b})$ and the choice of the r-dimensional affine space.

Reusing the notation from §3, let $S = \{\boldsymbol{p}_0^{(1)}, \ldots, \boldsymbol{p}_0^{(n)}, \boldsymbol{0}\}$, and let $\omega : S \to \mathbb{R}$ be the generic lifting function used for constructing the regular simplicial subdivision of $\operatorname{conv} S$ in §3. Without loss of generality, we can pick ω to have images only in $\mathbb{Q}$. With these, we introduce a new variable s and consider

$$H(\boldsymbol{t}, s) = \begin{cases} c_{11}\boldsymbol{t}^{\boldsymbol{p}_0^{(1)}} s^{\omega(\boldsymbol{p}_0^{(1)})} + \cdots + c_{1n}\boldsymbol{t}^{\boldsymbol{p}_0^{(n)}} s^{\omega(\boldsymbol{p}_0^{(n)})} - c_{10} s^{\omega(\boldsymbol{p}_0^{\boldsymbol{0}})} = \sum_{\boldsymbol{a}\in S} c_{1,\boldsymbol{a}}\boldsymbol{t}^{\boldsymbol{a}} s^{\omega(\boldsymbol{a})} \\ \qquad\qquad\qquad\qquad \vdots \\ c_{d1}\boldsymbol{t}^{\boldsymbol{p}_0^{(1)}} s^{\omega(\boldsymbol{p}_0^{(1)})} + \cdots + c_{dn}\boldsymbol{t}^{\boldsymbol{p}_0^{(n)}} s^{\omega(\boldsymbol{p}_0^{(n)})} - c_{d0} s^{\omega(\boldsymbol{0})} = \sum_{\boldsymbol{a}\in S} c_{d,\boldsymbol{a}}\boldsymbol{t}^{\boldsymbol{a}} s^{\omega(\boldsymbol{a})} \end{cases} \tag{25}$$

which is constructed by multiplying each term in (24) by a rational power of the new variable s whose exponent is determined by the lifting function $\omega : S \to \mathbb{Q}$. Clearly, $H(\boldsymbol{t}, 1) = \boldsymbol{0}$ is exactly the system (24) which we aim to solve (inside $(\mathbb{C}^*)^d$). As s varies, H represents a continuous deformation of the system (24), or a *homotopy*. The central idea behind the *homotopy continuation method* for solving systems of equations is the deformation of a system into a "starting system" which can be solved easily. Then numerical continuation methods are employed to trace the movement of the solutions of the starting system under the deformation toward the solutions of the original system which one aims to solve.

The key here is to find an appropriate starting system that can be easily solved. As is, $H(\boldsymbol{t}, 0)$ cannot be used as the starting system since at $s = 0$, the system is either identically zero or undefined. Therefore certain transformations are necessary to produce a meaningful and solvable starting system. Such transformations are given by the regular simplicial subdivision discussed in §3.

Let $\mathcal{D}$ be a regular simplicial subdivision obtained by the algorithm presented in §3.5. Recall that each cell in $\mathcal{D}$ is a projection of a cell of the form $\{\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_d\}$ such that $\operatorname{conv}\{\hat{\boldsymbol{a}}_0, \ldots, \hat{\boldsymbol{a}}_d\}$ is a lower d-face of $\operatorname{conv} \hat{S}$ that is characterized by (22). That is, there exists a (unique) vector of the form $\hat{\boldsymbol{\alpha}} = (\alpha_1, \ldots, \alpha_d, 1)$ such that

$$\begin{aligned} &\langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle = \langle \hat{\boldsymbol{a}}_j, \hat{\boldsymbol{\alpha}} \rangle \quad \text{for } j = 1, \ldots, d \\ &\langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle < \langle \hat{\boldsymbol{a}}, \hat{\boldsymbol{\alpha}} \rangle \quad \text{for all } \boldsymbol{a} \in S. \end{aligned} \tag{26}$$

Using $\hat{\boldsymbol{\alpha}} = (\alpha_1, \ldots, \alpha_d, 1)$, we shall consider the change of variables

$$\boldsymbol{t} = \begin{cases} t_1 = y_1 s^{\alpha_1} \\ \quad \vdots \\ t_d = y_d s^{\alpha_d} \end{cases} \tag{27}$$

with which H becomes

$$H(\boldsymbol{t}, s) = H(y_1 s^{\alpha_1}, \ldots, y_d s^{\alpha_d}, s) = \begin{cases} \sum_{\boldsymbol{a} \in S} c_{1,\boldsymbol{a}} \boldsymbol{y}^{\boldsymbol{a}} s^{\langle \boldsymbol{a}, \boldsymbol{\alpha} \rangle + \omega(\boldsymbol{a})} = \sum_{\boldsymbol{a} \in S} c_{1,\boldsymbol{a}} \boldsymbol{y}^{\boldsymbol{a}} s^{\langle \hat{\boldsymbol{a}}, \hat{\boldsymbol{\alpha}} \rangle} \\ \quad \vdots \\ \sum_{\boldsymbol{a} \in S} c_{d,\boldsymbol{a}} \boldsymbol{y}^{\boldsymbol{a}} s^{\langle \boldsymbol{a}, \boldsymbol{\alpha} \rangle + \omega(\boldsymbol{a})} = \sum_{\boldsymbol{a} \in S} c_{d,\boldsymbol{a}} \boldsymbol{y}^{\boldsymbol{a}} s^{\langle \hat{\boldsymbol{a}}, \hat{\boldsymbol{\alpha}} \rangle} \end{cases}$$

Let $\beta = \langle \hat{\boldsymbol{a}}_0, \hat{\boldsymbol{\alpha}} \rangle$ and define a new homotopy

$$H^{\boldsymbol{\alpha}, \beta}(\boldsymbol{y}, s) = s^{-\beta} H(y_1 s^{\alpha_1}, \ldots, y_d s^{\alpha_d}, s) = \begin{cases} s^{-\beta} \sum_{\boldsymbol{a} \in S} c_{1,\boldsymbol{a}} \boldsymbol{y}^{\boldsymbol{a}} s^{\langle \hat{\boldsymbol{a}}, \hat{\boldsymbol{\alpha}} \rangle} \\ \quad \vdots \\ s^{-\beta} \sum_{\boldsymbol{a} \in S} c_{d,\boldsymbol{a}} \boldsymbol{y}^{\boldsymbol{a}} s^{\langle \hat{\boldsymbol{a}}, \hat{\boldsymbol{\alpha}} \rangle} \end{cases} \tag{28}$$

Note that the new homotopy still has the necessary property that $H^{\boldsymbol{\alpha}, \beta}(\boldsymbol{y}, 1) = \boldsymbol{0}$ is identical to the system (24) which we aim to solve.

One important observation here is that, by (26), there are precisely $d+1$ terms in each component of $H^{\boldsymbol{\alpha}, \beta}(\boldsymbol{y}, s)$ having no power of s (the terms corresponding to $\boldsymbol{a}_0, \ldots, \boldsymbol{a}_d$), and all other terms have positive powers of s. Consequently, at $s = 0$, terms with positive powers of s vanish, leaving only

$$\begin{cases} c_{1,\boldsymbol{a}_0} \boldsymbol{y}^{\boldsymbol{a}_0} + c_{1,\boldsymbol{a}_1} \boldsymbol{y}^{\boldsymbol{a}_1} + \cdots + c_{1,\boldsymbol{a}_d} \boldsymbol{y}^{\boldsymbol{a}_d} = 0 \\ c_{2,\boldsymbol{a}_0} \boldsymbol{y}^{\boldsymbol{a}_0} + c_{2,\boldsymbol{a}_1} \boldsymbol{y}^{\boldsymbol{a}_1} + \cdots + c_{2,\boldsymbol{a}_d} \boldsymbol{y}^{\boldsymbol{a}_d} = 0 \\ \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ c_{d,\boldsymbol{a}_0} \boldsymbol{y}^{\boldsymbol{a}_0} + c_{d,\boldsymbol{a}_1} \boldsymbol{y}^{\boldsymbol{a}_1} + \cdots + c_{d,\boldsymbol{a}_d} \boldsymbol{y}^{\boldsymbol{a}_d} = 0 \end{cases} \tag{29}$$

To simplify the notation, let

$$C = \begin{bmatrix} c_{1,\boldsymbol{a}_0} & \cdots & c_{1,\boldsymbol{a}_d} \\ \vdots & \ddots & \vdots \\ c_{d,\boldsymbol{a}_0} & \cdots & c_{d,\boldsymbol{a}_d} \end{bmatrix} \qquad \Gamma = \begin{bmatrix} \boldsymbol{a}_0 & \cdots & \boldsymbol{a}_d \end{bmatrix}$$

then the above equation can be written as

$$C \cdot (\boldsymbol{y}^{\Gamma})^{\top} = \boldsymbol{0}. \tag{30}$$

For generic choices of the coefficients, there exists a nonsingular matrix $G \in M_{d \times d}(\mathbb{C})$ such that

$$GC = \begin{pmatrix} c_{11}^* & & & c_{12}^* \\ & c_{21}^* & & c_{22}^* \\ & & \ddots & \vdots \\ & & c_{d1}^* & c_{d2}^* \end{pmatrix},$$

for some $c_{ij}^* \in \mathbb{C}^*$. Then without altering its solution set, (30) can be transformed into the equivalent system

$$GC(\boldsymbol{t}^{\Gamma})^{\top} = \begin{cases} c_{11}^* \boldsymbol{y}^{\boldsymbol{a}_0} & & & + & c_{12}^* \boldsymbol{y}^{\boldsymbol{a}_d} & = & 0 \\ & c_{21}^* \boldsymbol{y}^{\boldsymbol{a}_1} & & + & c_{22}^* \boldsymbol{y}^{\boldsymbol{a}_d} & = & 0 \\ & & & \vdots & \vdots & \vdots & \\ & & c_{d1}^* \boldsymbol{y}^{\boldsymbol{a}_{d-1}} & + & c_{d2}^* \boldsymbol{y}^{\boldsymbol{a}_d} & = & 0 \end{cases} \tag{31}$$

which is also a Laurent binomial system. Therefore, the algorithm outlined can then be used to solve this system. The solutions are precisely the solutions of the starting system (29) for the homotopy $H^{\boldsymbol{\alpha},\beta}$. Then numerical continuation techniques can be applied to trace the solutions toward $s = 1$ producing solutions to the target system (24), which will be points in the witness set of the component V of $\mathcal{V}^*(\boldsymbol{x}^A - b)$.

Recall that the construction of the homotopy $H^{\boldsymbol{\alpha},\beta}$ depends on a cell in the regular simplicial subdivision $\mathcal{D}$ of $\operatorname{conv} S$. It is typical for $\mathcal{D}$ to contain more than one cell. In this case, each cell induces a different homotopy of the form of $H^{\boldsymbol{\alpha},\beta}$. The above construction is a special case of the *polyhedral homotopy* (Huber and Sturmfels, 1995), and its theory guarantees that as one goes through all the cells in $\mathcal{D}$, the resulting homotopies of the form $H^{\boldsymbol{\alpha},\beta}$ will find all the points in the witness set of V.

5. Related works

The approach developed in this article is a natural continuation of a rich web of works on the "mixed cell enumeration" problem initiated by the seminal work of Huber and Sturmfels (1995). The degree computation problem can be considered as a special case of the mixed cell enumeration problem known as the "unmixed case". Recent development in the computational aspects of this problem can be found in works such as Chen et al. (2014b), Gao and Li, (2000, 2003), Lee and Li (2011), Li and Li (2001), Mizutani and Takeda (2008), Mizutani et al. (2007), Verschelde et al. (1996). A broad survey of this topic can be found in Li (2003).

In particular, at the core of the proposed algorithm are the two complementary processes: the "extension" process which gradually extends lower faces one dimension at a time and the "pivoting" process which jumps directly from one cell to another. In the context of the DAG of "lower k-faces" (Fig. 3), the "extension" process explores the DAG vertically, and the "pivoting" process moves in the horizontal direction among the "lower d-faces".

The "extension" process is the core concept in several different mixed cell enumeration algorithms including Chen et al. (2014b), Gao and Li (2003), Lee and Li (2011), Li and Li (2001), Mizutani and Takeda (2008), Mizutani et al. (2007). The contribution of the present work to the development of the "extension" process is the adaptation to GPU devices. Unlike some of the previous attempts (e.g. Chen et al., 2014a, 2017) in using GPU devices in mixed cell enumeration algorithms where GPUs only perform certain specialized tasks (as an accelerator for simple linear algebra procedures), in the current work, however, the entire successive extension process runs on GPU devices.

Closely related to the present work is the article by Chen and Li (2014) which includes an empirical investigation on the use of existing (CPU-based) mixed cell enumeration algorithms in the computation of degrees of $\mathbb{C}^*$-solution set of binomial systems. Motivated by the encouraging results reported, the current work follows the same general approach of degree computation (as of computing normalized volume) but provides a significant improvement in both the absolute efficiency and parallel scalability by introducing the "pivoting" process into the algorithm. The original idea of the "pivoting" process in the context of "mixed cell enumeration" was proposed in Gao and Li (2000). However, in terms of efficiency, it was quickly eclipsed by algorithms based on the "extension" process, and there appears to be no further development of the idea between year 2000 and 2013.[5]

Moreover, in the present article, the "pivoting" and the "extension" processes are combined as we believe the complementary duo could offer much better scalability which is crucial in the GPU-based implementations. This is confirmed by the numerical experiments, to be presented in §6.

The graph-theoretic view of the "cell enumeration" process, adopted in this article, was introduced in Gao and Li (2000) and, independently, in Mizutani et al. (2007). The parallelization of the algorithm follows the same general idea attempted in Chen et al. (2014b), but it is modified, in this article, to adapt to the massively parallel GPU architectures.

[5] A recent work of Malajovich (2014), documents an independent development of an approach that appears to make use of a process similar to that proposed in Gao and Li (2000) and the current work. However, as the work focuses on the complexity analysis of algorithm, the authors are unable to directly assess the implications on performance or parallel scalability.

The homotopy (28) for computing witness sets, proposed in §4 is a special case of the polyhedral homotopy introduced in Huber and Sturmfels (1995) applied to the Laurent polynomial system (13).

6. Case study: master space of $\mathcal{N} = 1$ gauge theories

As a case study, we consider a system arising from theoretical physics, in particular, string theory, which has been studies in related works including Chen and Li (2014), Forcella et al. (2008b), Mehta et al. (2012). A central area of current research in string theory is the study of the vacuum moduli space, which, roughly speaking, is the space of continuous solutions (or the affine algebraic variety) of a multivariate nonlinear function, called the superpotential of the theory under consideration. Different positive-dimensional components of the vacuum moduli space correspond to different particle branches, such as mesonic, baryonic, etc. Symbolic algebraic geometry methods have been used to study the complicated structures of the vacuum moduli spaces of various string theory models (Gray, 2011; Gray et al., 2006, 2009). However, the methods are known to run out of the steam for even moderate sized systems due to the algorithmic complexity issues. Recently, numerical algebraic geometry methods have been introduced to string theory research and have solved bigger systems (Greene et al., 2013; Hauenstein et al., 2013; He et al., 2013; Martinez-Pedrera et al., 2013; Mehta, 2011; Mehta et al., 2012).

In this article, we consider special types of models coming from string theory in which the systems to be solved are binomial systems, and in which the vacuum moduli spaces are composed of unions of positive-dimensional components. We take a model which is actively investigated by string theorists because its vacuum moduli space is a combination of mesonic and baryonic branches (Forcella et al., 2008a, 2008b). Such moduli spaces are called *master spaces*.

In particular, we consider the superpotential for $\mathcal{N} = 1$ gauge theories for a $D3$-brane on the Abelian orbifold $\mathbb{C}^3/\mathbb{Z}_m \times \mathbb{Z}_k$. The superpotential for this theory, for fixed $m, k \in \mathbb{N}$, is given by

$$W_{m,k} = \sum_{i=0}^{m-1} \sum_{j=0}^{k-1} x_{i,j} y_{i+1,j} z_{i+1,j+1} - y_{i,j} x_{i,j+1} z_{i+1,j+1} \tag{32}$$

where the periodic boundary conditions are imposed, e.g., $x_{i,m} = x_{i,0}$ for any i and $x_{k,j} = x_{0,j}$ for any j. This is a polynomial in $3mk$ variables: $x_{i,j}$, $y_{i,j}$, $z_{i,j}$ for the combinations of $i \in \mathbb{Z}_k$ and $j \in \mathbb{Z}_m$. For example, when $m = k = 2$, the superpotential is

$$\begin{aligned} W_{2,2} = {} & x_{0,0} y_{1,0} z_{1,1} - y_{0,0} x_{0,1} z_{1,1} + x_{0,1} y_{1,1} z_{1,0} - y_{0,1} x_{0,0} z_{1,0} \\ & + x_{1,0} y_{0,0} z_{0,1} - y_{1,0} x_{1,1} z_{0,1} + x_{1,1} y_{0,1} z_{0,0} - y_{1,1} x_{1,0} z_{0,0}, \end{aligned}$$

a polynomial in 12 variables $x_{0,0}$, $x_{0,1}$, $x_{1,0}$, $x_{1,1}$, $y_{0,0}$, $y_{0,1}$, $y_{1,0}$, $y_{1,1}$, $z_{0,0}$, $z_{0,1}$, $z_{1,0}$, $z_{1,1}$. We are interested in finding the critical points of $W_{m,k}$, that is, points at which all the partial derivatives of the superpotential $W_{m,k}$, with respect to variables $x_{i,j}$, $y_{i,j}$, $z_{i,j}$, are zero. These points are precisely the solutions to the system of polynomial equation

$$\frac{\partial W_{m,k}}{\partial x_{i,j}} = \frac{\partial W_{m,k}}{\partial y_{i,j}} = \frac{\partial W_{m,k}}{\partial z_{i,j}} = 0 \tag{33}$$

in the variables $x_{i,j}$, $y_{i,j}$, $z_{i,j}$.

Notice that in $W_{m,k}$, each variable appears in exactly two distinct terms. Consequently, the partial derivative of $W_{m,k}$ with respect to each variable consists of exactly two terms, hence it forms a binomial polynomial. For instance,

$$\frac{\partial W_{2,2}}{\partial x_{0,0}} = y_{1,0} z_{1,1} - y_{0,1} z_{1,0}, \quad \frac{\partial W_{2,2}}{\partial x_{0,1}} = -y_{0,0} z_{1,1} + y_{1,1} z_{1,0}.$$

Therefore (33) is indeed a binomial system which shall simply be denoted by $\nabla W_{m,k}$. We are interested in computing the dimension and degree of components of the $\mathbb{C}^*$-solution set $\mathcal{V}^*(\nabla W_{m,k})$ of this system.

Table 1
The dimension of $\mathcal{V}^*(\nabla W_{m,k})$ for a range of values for m and k.

m/k	1	2	3	4	5	6	7	8
1	N/A	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18
3	5	8	11	14	17	20	23	26
4	6	10	14	18	22	26	30	34
5	7	12	17	22	27	32	37	42
6	8	14	20	26	32	38	44	50
7	9	16	23	30	37	44	51	58
8	10	18	26	34	42	50	58	66

Table 2
The dimension of $\mathcal{V}^*(\nabla W_{m,k})$ for a range of larger values for $m = k$.

$m = k$	9	10	11	12	13	14	15	20	25	30	35	40
Dim.	83	102	123	146	171	198	227	402	627	902	1227	1602

Table 3
The degree of the $\mathbb{C}^*$-solution set defined by (32) for a range of m and k values. This table lists only the results that can be computed within *1 hour* on a NVidia GTX 780 graphics card with the double-precision version of the GPU-based parallel algorithm presented in this work. Shaded entries correspond to and agree with the results already presented in Forcella et al. (2008b), Mehta et al. (2012). Certain entries are verified via CPU-based parallel algorithms (Chen and Li, 2014), but entries marked by * are results that cannot be computed with *any* CPU-based program within a reasonable amount of time (2 days for multi-core systems and 7 days for clusters). Entries marked by $\geq$ are lower bounds of the degrees computed by counting the total number of cells in the simplicial subdivision of the polytope associated with (32).

m/k	1	2	3	4	5	6	7	8
1		2	4	8	16	32	64	128
2	2	14	92	584	3632	22304	135872	823424
3	4	92	1620	26762	437038	7029180	111135118*	$\geq$ 100100328
4	8	584	26762	1169876	50467100	$\geq$ 11907022	$\geq$ 37567994	
5	16	3632	437038	50467100	$\geq$ 99710106	$\geq$ 62944504		
6	32	22304	7029180	$\geq$ 11907022	$\geq$ 62944504			
7	64	135872	111135118*	$\geq$ 37567994				
8	128	823424	$\geq$ 100100328					

The dimensions and the degrees of the top dimensional components of this system were first computed in Forcella et al. (2008b) for up to $m = 3$ and $k = 5$ using the Gröbner basis method. Later on, in Mehta et al. (2012), the dimensions and degrees of all the components for up to $m = 3 = k$ were carried out using numerical algebraic geometry methods. Table 1 and 2 show the dimension of $\mathcal{V}^*(\nabla W_{m,k}) \subset (\mathbb{C}^*)^{3mk}$ for a range of values for m and k which are computed via Smith Normal Form decomposition (5). More importantly, our implementation shows impressive efficiency in computing the degree of $\mathcal{V}^*(\nabla W_{m,k})$ for larger values of m and k, including a component of degree as high as 50467100, for $m = 4$ and $k = 5$. Table 3 shows the degree of $\mathcal{V}^*(\nabla W_{m,k})$ for a range of m and k values that can be computed within one hour[6] which is a significant expansion of the existing results presented in Forcella et al. (2008b), Mehta et al. (2012) (although the methods used in these two works yield more information than the proposed approach). In terms of efficiency and scalability, the

[6] The time limit is chosen based on two limitations of the experiment setup. Firstly, the GPU-devices used in the experiments (NVidia GTX 780) lacks ECC (error correcting code) support for its memory (known as DRAM) and is therefore prone to undetectable and catastrophic random error in stored data. The comprehensive study (Schroeder et al., 2011) conducted in real world scenarios concludes that "*DRAM error rates that are orders of magnitude higher than previously reported, with 25,000 to 70,000 errors per billion device hours per Mbit...*". While in the experiments we were not able to directly observe such errors, the surprisingly high estimate suggests that it can be risky to run long experiments on GPU-devices that have no ECC support. Moreover, for larger problems in this family that require longer running time, floating arithmetic in higher precision is likely needed which only has very limited support at this point.

Table 4
Speedup ratios achieved by the GPU-based double-precision (DP) and single-precision (SP) algorithms respectively on NVidia GTX 780 when compared to MixedVol-2.0 running on an Intel Xeon 2.4 GHz CPU. "0.00" represents speedup ratios too small to be measured reliably. The number of threads are chosen to be multiples of 32 which is the "warp size" (smallest group of threads in CUDA framework). Block size (number of threads working as a team) ranging from 64 to 1024 is used depending on the total number of threads. The GPU schedules thread execution based on a number of factors, therefore we cannot guarantee the simultaneous execution of all the threads.

GPU threads	DP speedup ratio	SP speedup ratio
64	0.00	0.00
128	0.00	0.00
256	0.00	0.00
512	0.91	0.73
1024	0.98	4.14
2048	1.15	6.66
4096	2.20	10.99
8192	4.01	18.71
2^{14}	7.99	35.00
2^{15}	15.00	40.10
2^{16}	16.33	45.33
2^{17}	29.47	44.99
2^{18}	28.33	41.06

Table 5
Speedup ratios achieved by using multiple identical NVidia GTX 780 devices with the single device performance (using the same algorithm) as a reference.

N.o. devices	1	2	3
Speedup over single device	100%	188%	213%
Max. speedup over CPU	28.33	54.12	61.00
Max. speedup over a cluster of 100 nodes	0.48	0.91	1.04

proposed algorithm is also a substantial improvement over existing algorithms investigated in Chen and Li (2014).

Table 4 shows the speedup ratio achieved by the GPU-based algorithm, presented in §3, over its closest serial CPU-based implementation MixedVol-2.0 (Lee and Li, 2011) which is widely regarded as one of fastest serial software program for computing "mixed volume" (see §5 and Appendix A for its connection with degree computation considered in this article). Remarkably, with sufficient GPU threads nearly 30 fold speedup ratio has been achieved by the double-precision version of the algorithm. When the single-precision version is used, even higher speedup ratio can be achieved. Unfortunately, it appears that single-precision is, in general, not reliable in handling very large problems due to its insufficient precision.

More important to note is the great potential of the GPU-based algorithm when multiple GPU devices are used. Table 5 shows the speedup ratio achieved by multiple GPU devices when compared to a single GPU, a single CPU, and a small cluster of 100 nodes. With three GPU devices, over 60 fold speedup over the single-threaded CPU-based algorithm (MixedVol-2.0) has been achieved. The most surprising result is the comparison between the GPU-based algorithm, developed in this article, running on three GPU devices and a similar CPU-based algorithm running on a small cluster. MixedVol-3 is a parallel version of MixedVol-2.0 (Lee and Li, 2011) and, now, a part of a larger software program Hom4PS-3 (Chen et al., 2014a). With three NVidia GTX 780 our GPU-based algorithm computes the degree for $\mathcal{V}^*(\nabla W_{4,5})$ faster than MixedVol-3 on a small cluster totaling 100 Intel Xeon 2.4 GHz processor cores.

In computing the degrees of $\mathcal{V}^*(\nabla W_{m,k})$ for certain larger m and k, while the GPU based algorithm was unable to compute the degree exactly due to insufficient numerical accuracy: In some cases the algorithm successfully obtained simplicial subdivisions of the polytopes associated with $\mathcal{V}^*(\nabla W_{m,k})$, but the volumes of cells, which are given as matrix determinants (17), could not be computed with

sufficient accuracy to ensure the exactness. In some other cases, the random lifting function fails to be sufficiently "generic". Consequently, the induced subdivision contains cells that are not simplices. In either cases, however, since the volume of each cell is at least one, the total number of cells is therefore a lower bound of the degree which equals the total volume of all the cells. Although these lower bounds are likely to be much smaller than the actual degrees, given the sheer size of these systems, these partial results still merit further investigations and improvements on the approach presented here. The lower bounds are therefore also included in Table 3 (entries marked with "$\geq$").

While the rigorous analysis and physical interpretation of the data presented here are outside the scope of this article, the rich set of data shown in Tables 1, 2, and 3 appear to show some general pattern. To motivate further research in this important problem, we summarize these patterns in the form of a conjecture:

Conjecture 8. *In general, for $m, k \in \mathbb{Z}^+$ with $m \neq k$, the solution set $\mathcal{V}^*(\nabla W_{m,k})$ consists of a single component of dimension*

$$\dim \mathcal{V}^*(\nabla W_{m,k}) = mk + 2.$$

Furthermore, for $m = 1$ and $m = 2$, the degree of the solution set is given by

$$\deg \mathcal{V}^*(\nabla W_{1,k}) = 2 \cdot \deg \mathcal{V}^*(\nabla W_{1,k-1}) = 2^{k-1}$$

$$\deg \mathcal{V}^*(\nabla W_{2,k}) = 6 \cdot \deg \mathcal{V}^*(\nabla W_{2,k-1}) + 2^{2k-3} = 2 \cdot 6^{k-1} + \sum_{j=0}^{k-2} 2^{2(k-j)-3} \cdot 6^j$$

7. Concluding remarks

In this article we proposed a parallel algorithm for computing the degree of components of a $\mathbb{C}^*$-solution set defined by a (Laurent) binomial system that is specifically designed for GPU devices. Rooted in the rich web of works on the mixed cell enumeration problem, the current contribution brings significant improvements over the existing methods in several different directions:

- Utilizing the special structure of binomial systems, this algorithm employs the "pivoting" process which was not used in the most efficient existing methods (Chen et al., 2014b; Gao and Li, 2003; Lee and Li, 2011; Lee et al., 2008; Li and Li, 2001; Mizutani and Takeda, 2008; Mizutani et al., 2007) (which depend on, exclusively, the "extension" process).
- With the goal of maximizing scalability, the two complementary processes "extension" and "pivoting" are combined via the use of a GPU managed hash table with minimum costs.
- In contrast to previous attempts, all computational intensive tasks are performed by GPUs in the proposed algorithm.
- Based on a formulation of the degree as the volume of a lattice polytope, the algorithm computes the degree without computing witness sets. In situations where witness sets are needed (for other operations in Numerical Algebraic Geometry), they can be computed by homotopies constructed as byproducts of the degree computation process. The benefit of this indirect approach is that the number of homotopy paths required is exactly the number of witness points.

Numerical experiments with the CUDA based implementation show remarkable performance and scalability in our case study of the $\mathcal{N} = 1$ gauge theories.

Acknowledgements

DM was supported by a DARPA Young Faculty Award and an Australian Research Council DECRA fellowship No. DE140100867. An internal preprint number for this article is ADP-15-2/T904. TC was supported in part by NSF under Grant DMS 11-15587. TC and DM would like to thank Daniel Brake, Yang-Hui He and Thomas Kahle for their feedback on this paper. TC would also like to thank

Dirk Colbry for the helpful discussions and the Institute for Cyber-Enabled Research at Michigan State University for providing the necessary hardware and computational infrastructure.

Appendix A. Kushnirenko/Bernshtein's theorem

Theorem 9 *(Kushnirenko, 1975; Bernshtein, 1975). Consider the system of k Laurent polynomial equations*

$$\begin{cases} c_{1,1}\boldsymbol{x}^{\boldsymbol{a}^{(1)}} + c_{1,2}\boldsymbol{x}^{\boldsymbol{a}^{(2)}} + \cdots + c_{1,k}\boldsymbol{x}^{\boldsymbol{a}^{(\ell)}} = 0 \\ c_{2,1}\boldsymbol{x}^{\boldsymbol{a}^{(1)}} + c_{2,2}\boldsymbol{x}^{\boldsymbol{a}^{(2)}} + \cdots + c_{2,k}\boldsymbol{x}^{\boldsymbol{a}^{(\ell)}} = 0 \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ c_{k,1}\boldsymbol{x}^{\boldsymbol{a}^{(1)}} + c_{k,2}\boldsymbol{x}^{\boldsymbol{a}^{(2)}} + \cdots + c_{k,k}\boldsymbol{x}^{\boldsymbol{a}^{(\ell)}} = 0 \end{cases}$$

in k variables $\boldsymbol{x} = (x_1, \ldots, x_k)$ in which every equation has the same set of monomials determined by exponent vectors $\boldsymbol{a}^{(1)}, \ldots, \boldsymbol{a}^{(\ell)} \in \mathbb{Z}^k$. For coefficients $c_{i,j} \in \mathbb{C}^*$ in a Zariski open and dense subset among all possible (complex) coefficients, the solutions of this system in $(\mathbb{C}^*)^k$ are all isolated and nonsingular. The total number of these isolated solutions is

$$k! \cdot \mathrm{Vol}_k(\mathrm{conv}\{\boldsymbol{a}^{(1)}, \ldots, \boldsymbol{a}^{(\ell)}\}).$$

In Bernshtein (1975), this theorem takes a significantly more general form where the number of isolated nonzero $\mathbb{C}^*$-solutions of a system of a Laurent polynomial system is shown to be equal to the *mixed volume* of the *Newton polytopes* of the system. This number is known as the *BKK bound* of a Laurent polynomial system. Therefore, the degree computation discussed above can be considered as a special case of the BKK bound i.e. mixed volume computation.

Appendix B. Transformation matrices used in Smith Normal Form

In the Smith Normal Form decomposition $A = P^{-1}\left[\begin{smallmatrix} D & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right] Q^{-1}$ the transformation matrices P and Q are not unique. Suppose there is a different pair of unimodular matrices $\tilde{P} = \left[\begin{smallmatrix} \tilde{P}_r \\ \tilde{P}_0 \end{smallmatrix}\right] \in M_{n\times n}(\mathbb{Z})$ where $\tilde{P}_0 \in M_{(n-r)\times n}(\mathbb{Z})$ and $\tilde{Q} \in M_{m\times m}(\mathbb{Z})$ such that

$$\begin{bmatrix} \tilde{P}_r \\ \tilde{P}_0 \end{bmatrix} A \begin{bmatrix} \tilde{Q}_r & \tilde{Q}_0 \end{bmatrix} = \begin{bmatrix} D & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Then we must have

$$\tilde{P}_0 A = \tilde{P}_0 P^{-1} \begin{bmatrix} D & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} Q^{-1} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

Recall that both P and Q are nonsingular, therefore

$$\tilde{P}_0 P^{-1} = \begin{bmatrix} \mathbf{0} & G \end{bmatrix}$$

for some matrix $G \in M_{(n-r)\times(n-r)}(\mathbb{Z})$, and hence

$$\tilde{P}_0 = \begin{bmatrix} \mathbf{0} & G \end{bmatrix} P = \begin{bmatrix} \mathbf{0} & G \end{bmatrix} \begin{bmatrix} P_r \\ P_0 \end{bmatrix} = G P_0.$$

Moreover, since P_0 and $\tilde{P}_0$ consists of the $n-r$ rows of the unimodular matrices P and $\tilde{P}$ respectively and hence must be of full rank, the matrix G must be nonsingular. Also note that

$$\pm 1 = \det \tilde{P} P^{-1} = \det \begin{bmatrix} \tilde{P}_r P^{-1} \\ \tilde{P}_0 P^{-1} \end{bmatrix} = \begin{bmatrix} \tilde{P}_r P^{-1} & \\ \mathbf{0} & G \end{bmatrix}$$

therefore G must be unimodular. In other words, columns of $\tilde{P}_0$ must be the images of columns of P_0 under the linear transformation given by the unimodular matrix G.

Appendix C. Parallel Smith Normal Form computation

As summarized in Proposition 1 and Proposition 3, the computation of Smith Normal Form of the exponent matrix A in the binomial system $\boldsymbol{x}^A - \boldsymbol{b}$ is a crucial preprocessing step for the algorithm proposed in this article.

One of the classic algorithms for computing the Smith Normal Form uses successive row and column reductions (see e.g., Cohen, 1993, Algorithm 2.4.14): Consider the special case where $A = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$ with a_1, a_2 nonzero, that is, take $n = 2$ and $m = 1$. By Bézout's identity, there exist s and t such that $d := \gcd(a_1, a_2) = s\,a_1 + t\,a_2$. Let

$$P = \begin{bmatrix} s & t \\ -\frac{a_2}{d} & \frac{a_1}{d} \end{bmatrix},$$

then $\det P = \frac{sa_1 + ta_2}{d} = \frac{d}{d} = 1$ and

$$PA = \begin{bmatrix} s & t \\ -\frac{a_2}{d} & \frac{a_1}{d} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} sa_1 + ta_2 \\ -\frac{a_2 a_1}{d} + \frac{a_2 a_2}{d} \end{bmatrix} = \begin{bmatrix} d \\ 0 \end{bmatrix}$$

Similarly, for the special case $A = [\,a_1\ a_2\,]$, let $Q = \begin{bmatrix} s & \frac{-a_2}{d} \\ t & \frac{a_1}{d} \end{bmatrix}$, then $A\,Q = \begin{bmatrix} d & 0 \end{bmatrix}$. In general, $n \times n$ and $m \times m$ version of the above matrices P and Q can be constructed to perform row and column reduction respectively for a $n \times m$ integer matrix.

It can be shown that using repeated row and column reduction together with potential row and column permutations one can construct unimodular matrices $P^{(1)}, \ldots, P^{(k)} \in M_{n\times n}(\mathbb{Z})$ and $Q^{(1)}, \ldots, Q^{(\ell)} \in M_{m\times m}(\mathbb{Z})$ such that

$$P^{(k)} \cdots P^{(1)}\, A\, Q^{(1)} \cdots Q^{(\ell)} = \begin{pmatrix} d_1 & & & & \\ & \ddots & & & \\ & & d_r & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0 \end{pmatrix}$$

with $r = \operatorname{rank} A$ and $d_1, \ldots, d_r$ nonzero.

References

Bárány, I., Füredi, Z., 1987. Computing the volume is difficult. Discrete Comput. Geom. 2 (1), 319–326.

Bernshtein, D.N., 1975. The number of roots of a system of equations. Funct. Anal. Appl. 9 (3), 183–185.

Büeler, B., Enge, A., Fukuda, K., 2000. Exact volume computation for polytopes: a practical study. In: Kalai, G., Ziegler, G.M. (Eds.), Polytopes — Combinatorics and Computation. In: DMV Seminar, vol. 29. Birkhäuser, Basel, pp. 131–154.

Chen, T., Lee, T.-L., Li, T.-Y., 2014a. Hom4ps-3: a parallel numerical solver for systems of polynomial equations based on polyhedral homotopy continuation methods. In: Hong, H., Yap, C. (Eds.), Mathematical Software – ICMS 2014. In: Lecture Notes in Computer Science, vol. 8592. Springer, Berlin, Heidelberg, pp. 183–190.

Chen, T., Lee, T.-L., Li, T.-Y., 2014b. Mixed volume computation in parallel. Taiwan. J. Math. 18 (1), 93–114.

Chen, T., Lee, T.-L., Li, T.-Y., 2017. Mixed cell computation in Hom4PS-3. In: Numerical Algebraic Geometry – Special Issue. J. Symb. Comput. 79, 516–534.

Chen, T., Li, T.-Y., 2014. Solutions to systems of binomial equations. Ann. Math. Sil. 28, 7–34.

Cohen, H., 1993. A Course in Computational Algebraic Number Theory, vol. 138. Springer.

Cox, D.A., Little, J.B., Schenck, H.K., 2011. Toric Varieties. American Mathematical Soc.

Eisenbud, D., Sturmfels, B., 1996. Binomial ideals. Duke Math. J. 84 (1), 1–46.

Forcella, D., Hanany, A., He, Y.-H., Zaffaroni, A., 2008a. Mastering the master space. Lett. Math. Phys. 85, 163–171.

Forcella, D., Hanany, A., He, Y.-H., Zaffaroni, A., 2008b. The master space of $\mathcal{N} = 1$ gauge theories. J. High Energy Phys. 0808, 012.

Fulton, W., 1993. Introduction to Toric Varieties. Annals of Mathematics Studies, vol. 131. Princeton University Press.

Fulton, W., 1998. Intersection Theory. Springer, New York.

Gao, T., Li, T.-Y., 2000. Mixed volume computation via linear programming. Taiwan. J. Math. 4 (4), 599–619.

Gao, T., Li, T.-Y., 2003. Mixed volume computation for semi-mixed systems. Discrete Comput. Geom. 29 (2), 257–277.

Gelfand, I.M., Kapranov, M.M., Zelevinsky, A.V., 1994. Discriminants, Resultants, and Multidimensional Determinants. Mathematics: Theory & Applications. Birkhäuser, Boston.
Gray, J., 2011. A simple introduction to Grobner basis methods in string phenomenology. Adv. High Energy Phys. 2011, 217035.
Gray, J., He, Y.-H., Ilderton, A., Lukas, A., 2009. STRINGVACUA: a Mathematica package for studying vacuum configurations in string phenomenology. Comput. Phys. Commun. 180, 107–119.
Gray, J., He, Y.-H., Lukas, A., 2006. Algorithmic algebraic geometry and flux vacua. J. High Energy Phys. 0609, 031.
Greene, B., Kagan, D., Masoumi, A., Mehta, D., Weinberg, E.J., Xiao, X., 2013. Tumbling through a landscape: evidence of instabilities in high-dimensional moduli spaces. Phys. Rev. D 88 (2), 026005.
Hartshorne, R., 1977. Algebraic Geometry. Graduate Texts in Mathematics, vol. 52. Springer.
Hauenstein, J., He, Y.-H., Mehta, D., 2013. Numerical elimination and moduli space of vacua. J. High Energy Phys. 1309, 083.
He, Y.-H., Mehta, D., Niemerg, M., Rummel, M., Valeanu, A., 2013. Exploring the potential energy landscape over a large parameter-space. J. High Energy Phys. 1307, 050.
Herlihy, M., Shavit, N., 2012. The Art of Multiprocessor Programming. Elsevier.
Huber, B., Sturmfels, B., 1995. A polyhedral method for solving sparse polynomial systems. Math. Comput. 64 (212), 1541–1555.
Kahle, T., 2010. Decompositions of binomial ideals. Ann. Inst. Stat. Math. 62 (4), 727–745.
Kahle, T., Miller, E., 2014. Decompositions of commutative monoid congruences and binomial ideals. Algebra Number Theory 8 (6), 1297–1364.
Kushnirenko, A.G., 1975. A Newton polyhedron and the number of solutions of a system of k equations in k unknowns. Usp. Math. Nauk 30, 266–267.
Lee, T.-L., Li, T.-Y., 2011. Mixed volume computation in solving polynomial systems. Contemp. Math. 556, 97–112.
Lee, T.-L., Li, T.-Y., Tsai, C.-H., 2008. HOM4ps-2.0: a software package for solving polynomial systems by the polyhedral homotopy continuation method. Computing 83 (2), 109–133.
Li, T.-Y., 2003. Numerical solution of polynomial systems by homotopy continuation methods. In: Ciarlet, P.G. (Ed.), Handbook of Numerical Analysis, vol. 11. North-Holland, pp. 209–304.
Li, T.-Y., Li, X., 2001. Finding mixed cells in the mixed volume computation. Found. Comput. Math. 1 (2), 161–181.
Loera, J.D., Rambau, J., Santos, F., 2010. Triangulations: Structures for Algorithms and Applications. Springer Science & Business Media.
Malajovich, G., 2014. Computing mixed volume and all mixed cells in quermassintegral time. arXiv:1412.0480 [math].
Martinez-Pedrera, D., Mehta, D., Rummel, M., Westphal, A., 2013. Finding all flux vacua in an explicit example. J. High Energy Phys. 1306, 110.
Mehta, D., 2011. Numerical polynomial homotopy continuation method and string vacua. Adv. High Energy Phys. 2011, 263937.
Mehta, D., He, Y.-H., Hauenstein, J.D., 2012. Numerical algebraic geometry: a new perspective on string and gauge theories. J. High Energy Phys. 1207, 018.
Miller, E., Sturmfels, B., 2005. Combinatorial Commutative Algebra, vol. 227. Springer.
Mizutani, T., Takeda, A., 2008. DEMiCs: a software package for computing the mixed volume via dynamic enumeration of all mixed cells. In: Stillman, M., Verschelde, J., Takayama, N. (Eds.), Software for Algebraic Geometry. In: The IMA Volumes in Mathematics and Its Applications, vol. 148. Springer, pp. 59–79.
Mizutani, T., Takeda, A., Kojima, M., 2007. Dynamic enumeration of all mixed cells. Discrete Comput. Geom. 37 (3), 351–367.
NVIDIA Corporation, 2011. NVIDIA CUDA C programming guide. Technical report.
Schroeder, B., Pinheiro, E., Weber, W.-D., 2011. DRAM errors in the wild: a large-scale field study. Commun. ACM 54 (2), 100–107.
Skiena, S.S., 2009. The Algorithm Design Manual. Springer Science & Business Media.
Smith, H.J.S., 1861. On systems of linear indeterminate equations and congruences. Philos. Trans. R. Soc. Lond. 151, 293–326.
Sommese, A.J., Wampler, C.W., 1996. Numerical algebraic geometry. In: The Mathematics of Numerical Analysis. In: Lectures in Applied Mathematics, vol. 32. AMS, pp. 749–763.
Sommese, A.J., Wampler, C.W., 2005. The Numerical Solution of Systems of Polynomials Arising in Engineering and Science. World Scientific Pub. Co., Inc.
Sturmfels, B., 1997. Equations defining toric varieties. In: Proc. Symp. Pure Math. American Mathematical Society.
Verschelde, J., Gatermann, K., Cools, R., 1996. Mixed-volume computation by dynamic lifting applied to polynomial system solving. Discrete Comput. Geom. 16 (1), 69–112.