ISBN: 978-3-95450-168-7 doi:10.18429/JACoW-IPAC2015-MOPJE039

GENERALISED TRUNCATED POWER SERIES ALGEBRA FOR FAST PARTICLE ACCELERATOR TRANSPORT MAPS

L. Deniau, C. I. Tomoiagă, CERN, Geneva, Switzerland

Abstract

A new Generalised Truncated Power Series Algebra (GTPSA) has been developed for extending, simplifying and optimising the transport maps used by particle accelerator simulation codes. TPSA are intensively used in optics code to describe transport maps of the elements constituting the particle accelerator to any order. GTPSA extend the degrees to inhomogeneous ones, where separate degrees can be specified for each variable and constrained by two total orders, one for map variables and one for ordinary variables. This allows tracking inhomogeneous planes of the 6D phase space with many extra variables. A complete set of new formulas and data structures have been derived to address the problem of memory consumption required for efficient computation of high order TPSA, including generalised indexing, multiplication and composition of inhomogeneous multivariate polynomials. The implementation has been benchmarked against well established libraries for the common subset with TPSA, and outperforms all of them for supported differential algebra operators on low and high orders, and high number of variables.

INTRODUCTION

TPSA packages are intensively used in optics code to describe transport maps of the elements constituting the particle accelerator to any order [1, 2]. Most of the known fast multiplication algorithms working for univariate polynomials do not straightforwardly extend to several variables [3], and therefore the brute force approach remains the most efficient one for *truncated* power series. The performance improvements have to come from other kind of optimizations [4–7].

Notations

The TPSA are multivariate polynomials $P(\vec{z}) \in \mathbb{K}[\vec{z}]$, $\vec{z} = \{z_1, ..., z_n\}$ truncated at order d, where the commutative ring is generally $\mathbb{R}$ or $\mathbb{C}$. A multivariate polynomial is defined by the sum of all the monomials of order d or less:

$$P(\vec{z}) = \sum_{i=1}^{N} a_i \prod_{j=1}^{n} z_j^{\alpha_{ij}}, \quad \sum_{j=1}^{n} \alpha_{ij} \leq d, \tag{1}$$

where the number N of coefficients a_i is given by:

$$N = C(n+d, n) = C(n+d, d) = \frac{(n+d)!}{n!d!}. \tag{2}$$

The binomial coefficient C is symmetric in variables and orders, hence a TPSA package can deal equivalently with high orders and low number of variables or low orders and high number of variables. The number N_i of monomials of order i with $0 \leq i \leq d$ or equivalently of monomials in the variable z_j^{d-i} with $1 \leq j \leq n$ can be calculated from the recurrence property of the binomial coefficient:

$$\begin{aligned} N_i &= C(n+i, n) - C(n+i-1, n) \\ &= C(n+i, n-1). \end{aligned} \tag{3}$$

Representations

The (unordered) sum of the monomials in (1) can be ordered in two practical forms:

1. By the **product of univariate polynomials** in the variables z_i, which is used by the indexing function:

$$P(\vec{z}) = \prod_{i=1}^{n} P(z_i) = \sum_{i=0}^{d} \sum_{j=1}^{N_{d-i}} a_{ij} \vec{z}_{ij}. \tag{4}$$

The monomial $\vec{z}_{ij}$ associated with the coefficient a_{ij} are generated recursively from $z_k \times \{z_{k+1}, ..., z_n\}$, e.g. the outer sum over i corresponds to the variable z_1. As an example, a multivariate polynomial of order 3 with variables $\{z_1, z_2, z_3\}$ can be represented by the series:

$$\begin{aligned} P(\vec{z}) =& a_1 z_1^0 z_2^0 z_3^0 + \\ & a_2 z_1^0 z_2^0 z_3^1 + a_3 z_1^0 z_2^0 z_3^2 + a_4 z_1^0 z_2^0 z_3^3 + \\ & a_5 z_1^0 z_2^1 z_3^0 + a_6 z_1^0 z_2^1 z_3^1 + a_7 z_1^0 z_2^1 z_3^2 + \\ & ... + a_{20} z_1^3 z_2^0 z_3^0. \end{aligned} \tag{5}$$

2. By the **sum of homogeneous multivariate polynomials** in the orders $i = \sum_{k=1}^{n} \alpha_{ik}$, , which is used by the storage of the a_{ij} and by the optimized multiplication:

$$P(\vec{z}) = \sum_{i=0}^{d} P_i(\vec{z}) = \sum_{i=0}^{d} \sum_{j=1}^{N_i} a_{ij} \vec{z}_{ij}. \tag{6}$$

The monomial $\vec{z}_{ij}$ associated with the coefficient a_{ij} are generated recursively from $P_1 \times P_{i-1}$. As an example, a multivariate polynomial of order 3 with variables $\{z_1, z_2, z_3\}$ can be represented by the series:

$$\begin{aligned} P(\vec{z}) =& a_1 z_1^0 z_2^0 z_3^0 + \\ & a_2 z_1^0 z_2^0 z_3^1 + a_3 z_1^0 z_2^1 z_3^0 + a_4 z_1^1 z_2^0 z_3^0 + \\ & a_5 z_1^0 z_2^0 z_3^2 + a_6 z_1^0 z_2^1 z_3^1 + a_7 z_1^0 z_2^2 z_3^0 + \\ & ... + a_{20} z_1^3 z_2^0 z_3^0. \end{aligned} \tag{7}$$

Generalization

The GTPSA extends the TPSA by splitting the variables in two kinds $\vec{z} = \{\vec{x}, \vec{k}\}$, where the **map variables** $\vec{x} = \{x_1, ..., x_v\}$ with $v \geq 1$ are present in both the map and the (G)TPSA of the map variables, and where the **knob variables** $\vec{k} = \{k_{v+1}, ..., k_n\}$ with $n - v \geq 0$ are only present in the GTPSA of the map variables.

ISBN: 978-3-95450-168-7 doi:10.18429/JACoW-IPAC2015-MOPJE039

This distinction comes from the physical meaning of these variables, where for example $\partial x_i/\partial k_j$ is the linear variation of the coordinate x_i (e.g. particle position) versus the knob k_j (e.g. magnet strength), while $\partial k_j/\partial x_j$ has no physical meaning (e.g. magnet strength vs. particle position). Hence, the knobs are never map variables.

The GTPSA also extends the TPSA by allowing to specify a maximum order d_j for each variable z_j as well as two total orders d_x and d_k for each kind of variables:

$$0 \leq \alpha_{ij} \leq d_j, \quad \sum_{j=1}^{v} \alpha_{ij} \leq d_x \leq \sum_{j=1}^{v} d_j, \tag{8}$$

$$d_k \leq d_x, \quad \sum_{j=v+1}^{n} \alpha_{ij} \leq d_k \leq \sum_{j=v+1}^{n} d_j. \tag{9}$$

Map variables x_j can have zero maximum order, which corresponds to scalars. In this case they do not appear in the (G)TPSAs of the map variables, only as map variables. On the other hand knobs cannot be specified with zero maximum order.

The complexity behind these generalizations is twofold: to take care of the unallowed monomials in (3), (4) and (6) due to non-uniform maximum orders, and to handle all combinations of operations between scalars and heterogeneous GTPSA, including the truncation of the calculations to the specific orders of left-hand-side map variables of the equations of motion to save unneeded high order computations.

Specifications

The GTPSA extensions are useful to unify different kind of studies using the same equations of motion, e.g. 4D and 6D beam dynamics. The maps are specified by the variables maximum order d_j and the total orders d_x and d_k. The following specifications for maps of GTPSA are of particular interest, assuming a 6D phase space $\vec{x} = \{x, p_x, y, p_y, s, p_s\}$:

- $\vec{x} = \{0, 0, 0, 0, 0, 0\}$ corresponds to 6D zero order particle tracking, i.e. particles orbits, where the map is simply a vector of six scalars. In this case, $d_x = 0$, $d_k = 0$ and $\vec{k} = \vec{0}$.
- $\vec{x} = \{2, 2, 2, 2, 0, 0\}$ corresponds to 6D with second order transverse beam dynamics and zero order longitudinal beam dynamics, which emulates 4D beam dynamics. In this case, $d_x = 2$, $d_k = 0$ and $\vec{k} = \vec{0}$.
- $\vec{x} = \{1, 1, 1, 1, 4, 4\}$ corresponds to 6D with first order transverse beam dynamics and fourth order longitudinal beam dynamics. In this case, $d_x = 4$, $d_k = 0$ and $\vec{k} = \vec{0}$, meaning that mixed high order terms like $\partial^4 p_s/(\partial x \partial p_x \partial y \partial p_y)$ exist.
- $\vec{x} = \{1, 1, 1, 1, 0, 0\}$ and $\vec{k} = \{1, ..., 1\}$ corresponds to 6D with first order transverse beam dynamics and zero order longitudinal beam dynamics with $n - v$ (few hundreds) first order knobs in the GTPSA (e.g. strength of orbit correctors). In this case, $d_x = 1$ and $d_k = 1$, meaning that terms like $\partial x/\partial k_j$ and $\partial y/\partial k_j$ exist and can be used directly by orbit correction algorithms.

All these inhomogeneous cases use the same equations of motion for each kind of element to track the particle through the lattice. The orders and the complexity of the calculations are automatically and optimally handled by the GTPSA.

INDEXING

The indexing function is one of the key features of the GTPSA as it needs to efficiently discard unallowed monomials by (8) and (9), and calculates the index of the allowed monomials. The indexing function for TPSA is simple as soon as the TPSA are ordered by variables (i.e. from (4)). The unique TPSA index t_i of the ith-monomial can be calculated from the monomial orders $\{\alpha_{ij}\}$ using:

$$t_i = \sum_{j=1}^{n} H(j, s_j) - H(j, s_{j+1}), \quad s_j = \sum_{k \geq j}^{n} \alpha_{ik}, \tag{10}$$

where the matrix H of size $n \times d$ is given by the equations:

$$H(j, 0) = 0, \qquad j = 1, ..., n \tag{11}$$

$$H(1, i) = i, \qquad i = 0, ..., d \tag{12}$$

$$H(j, 1) = H(j - 1, d_{j-1} + 1), \tag{13}$$

$$H(j, i) = H(j, 1) + H(j, i - 1) - H(j - 1, i - 1) \tag{14}$$

For example, given a TPSA specified by $n = 3$ and $d = 3$, the matrix H looks like:

(j,i)	0	1	2	3
1	0	1	2	3
2	0	4	7	9
3	0	10	16	19

(15)

The index t_i can be trivially translated to the index p_i of a TPSA ordered by homogeneous polynomials by a one-to-one lookup table T going from (4) to (6), that is $p_i = T[t_i]$.

Generalization

In the case of the GTPSA, the matrix H becomes sparse and therefore the recurrent equations (13) and (14) cannot be used to build H, but of course (10), (11), and (12) are still valid. The process of building H has **two phases**.

The first phase initializes H with the indexes where each variable changes of order in (4). For example, with $\vec{x} = \{1, 1, 3, 1\}$ and $d_x = 4$, this phase initializes H with:

(j,i)	0	1	2	3	4
1	0	1	2	3	4
2	0	2	4	·	·
3	0	4	8	12	15
4	0	15	·	·	·

(16)

The row $H(3, .)$ indicates that x_3 gets orders $\{1, 2, 3, 4\}$ at indexes $\{4, 8, 12, 15\}$ while x_4 gets order 1 at index 15. Note that x_3 can reach the mixed order 4 when combined with other variables due to $d_x = 4$ in (10). The dots represent unknown monomials remaining after the first phase.

The second phase solves the system of non-linear equations that replaces (14) not anymore valid for the GTPSA. In order to quickly find the unknowns of H, the solver linearizes the problem by doing the following steps:

ISBN: 978-3-95450-168-7 doi:10.18429/JACoW-IPAC2015-MOPJE039

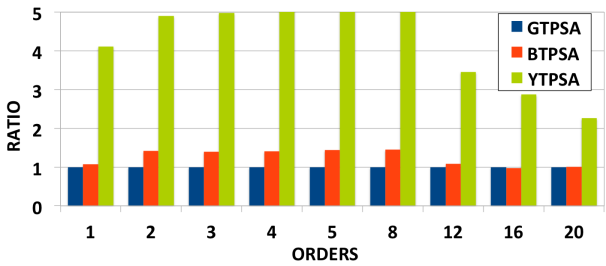

Figure 1: Relative performance of the indexing functions.

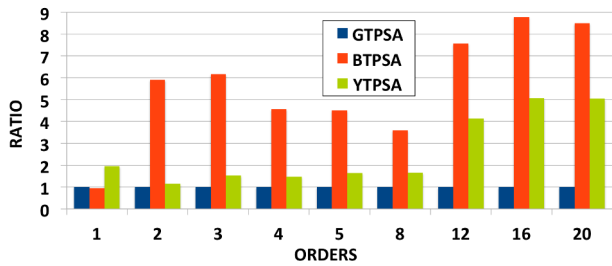

Figure 2: Relative performance of the multiplications.

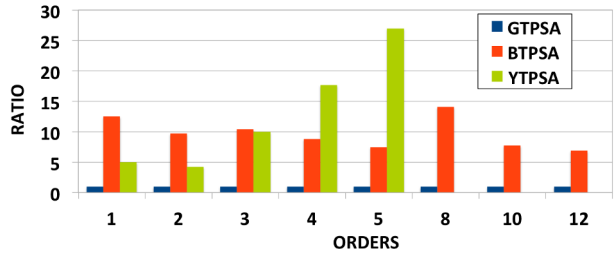

Figure 3: Relative performance of the compositions.

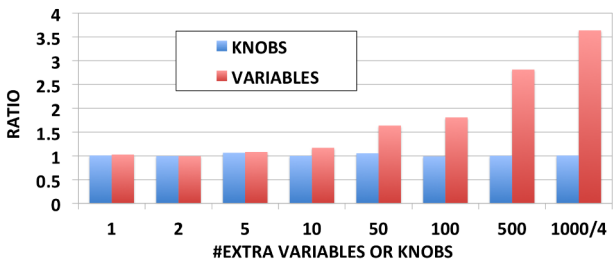

Figure 4: Relative performance of the knobs (inhomogeneous) vs. variables (homogeneous) for $d_x = 2$ and $d_k = 1$.

1. For each unknown taken by increasing order in H,
2. Generate a monomial that includes the unknown,
3. Get the index from (10) using the incomplete H,
4. Find the correct index using representation of (4),
5. The solution is the difference between the two indexes.

This procedure will fill the unknown slots of H in (16) as follows, where the dots are now the unallowed monomials:

(j,i)	0	1	2	3	4
1	0	1	2	3	4
2	0	2	4	6	8
3	0	4	8	12	15
4	0	15	·	·	·

(17)

MULTIPLICATION, COMPOSITION

The multiplication of GTPSA is not very different from TPSA. It uses the homogeneous polynomials representation of (6), which is the most efficient for multiplying *truncated* multivariate polynomials. The only major difference is that the indexing function for the resulting monomial indexes is precomputed using (10) and the lookup table T to obtain the indexes of (6). These precomputed "destination" indexes are stored in lookup matrices L_{ij} corresponding to each combination of multiplication $R_{i+j} = P_i \times Q_j$ with $i \leq j$, and where a special negative index is used for tagging unallowed monomials. This saves a huge amount of memory compared to table handling indexing of the entire TPSA at once as in the Yang TPSA [4], and increase the number of variables and/or the high orders supported by the GTPSA.

Optimization

The multiplication of truncated multivariate polynomials is the key operation of the GTPSA, and requires careful optimization. GTPSA takes care of the following three cases that lead to specific optimized multiplications:

- symmetric: $P_i \times Q_j$ and $P_j \times Q_i$ both exist,
- asymmetric: either $P_i \times Q_j$ or $P_j \times Q_i$ exists,
- triangular: $P_i \times Q_i$ exists (diagonal).

The representation by homogeneous polynomials offers also to parallelize these multiplications per resulting orders. The map composition is optimal as it performs $v \times N$ multiplications. For orders ≥ 12 (resp. order ≥ 6 for composition), we *balance the load* of the calculation over the available CPU cores, and we split the highest order on two cores, as it represents by itself about half of the total calculation.

BENCHMARK

The relative performances of the indexing function, the multiplication, and the map composition, all with 6 variables and no knobs for compatibility with TPSA, (i.e. $n = v = 6$), are presented in Figs. 1, 2 and 3, where a ratio of e.g. 4 means $\times 4$ **slower** than the GTPSA. These results show that the GPTSA outperforms the Berz TPSA [1] and the Yang TPSA [4] (resp. BTPSA and YTPSA) despite of its extra complexity and its extended flexibility. The Fig. 4 shows the speed improvement of the multiplication using GTPSA that allows 6 variables at order 2 and many extra knobs at order 1 compared to TPSA where everything must be specified as variables at order 2.

CONCLUSIONS

We have presented a new Generalized TPSA package that offers much more flexibility for the computation of maps in beam dynamics and complex maps for normal form analysis. It outperforms well-known high performance packages while adding extra flexibility and in spite of complexity. Future work will focus on optimizing sparse maps using a mixed representation of dense and sparse homogeneous polynomials, which should be more efficient for elements maps.

ISBN: 978-3-95450-168-7 doi:10.18429/JACoW-IPAC2015-MOPJE039

REFERENCES

[1] M. Berz, *Modern Map Methods in Particle Beam Physics*, Academic Press, 1999, http://bt.pa.msu.edu/pub.

[2] E. Forest, *Beam Dynamics*, CRC Press, 1998.

[3] J. van der Hoeven and G. Lecerf, *On the Complexity of Multivariate Blockwise Polynomial Multiplication*, ISSAC'12, Grenoble, France, 2012.

[4] L. Yang, *Array Based Truncated Power Series Package*, IPAC'09 (THPSC059), San Francisco, USA, 2009.

[5] J.F. Ostiguy and L. Michelotti, *MXYZPTLK: An Efficient, Native, C++ Differentiation Engine*, PAC'07 (THPAN113), Albuquerque, New Mexico, USA, 2007.

[6] J. R. Cary and S. G. Shasharina, *Efficient C++ Library for Differential Algebra*, EPAC'98 (THP38C), Stockholm, Sweden, 1998.

[7] M. Gastineau, *Parallel Operations of Sparse Polynomials on Multicores - I. Multiplication and Poisson Bracket*, PASCO'10, Grenoble, France, 2010.