# ZLIB: A NUMERICAL LIBRARY FOR DIFFERENTIAL ALGEBRA AND LIE ALGEBRAIC TREATMENT OF BEAM DYNAMICS

Yiton T. Yan
Superconducting Super Collider Laboratory
2550 Beckleymeade Avenue, Dallas, TX 75237

*Abstract*

A dynamic-memory numerical library, "ZLIB", has been developed in an attempt to offer efficient numerical routines on supercomputers for differential algebra and the relevant Lie algebraic techniques for mapping studies of beam dynamics. Optimization of ZLIB has been attempted for vector computing as well as scalar computing. Parallel computing (multi-tasking) can also be easily employed. Currently, ZLIB contains more than 200 subroutines.

## I. Introduction

With limited computer memory, and limited computational speed, differential algebra should be treated as the algebra of truncated power series [1]. The algebra of low order truncated power series can be easily accomplished with a simple data structure. However, in most cases, high-order truncated power series is desirable. Therefore, a special data structure is necessary to optimize both the allocation of the computer memory and the numerical speed.

"ZLIB" [2] has been developed for differential algebra, mainly for use on supercomputers. The use of "ZLIB" is similar to the use of the "IMSL" library with an additional feature that the internal working memories are dynamically generated. Routines in "ZLIB" are vectorized and can be multi-tasked easily for supercomputers. There are two sub-libraries in "ZLIB", the "TPALIB" and the "ZPLIB", with unique data structures. The "TPALIB" is more flexible in dealing with a different number of variables, and therefore is more suitable for use in extracting a one-turn (or one-period) map for a storage ring such as the SSC. Indeed, this sublibrary has been used for a program named "Zmap" [3] to extract a one-turn map for the systematic circular accelerator program Teapot [4]. The "ZPLIB" is more flexible in dealing with a different number of orders, and therefore is more suitable for use in analyzing a map. Lie algebraic routines are mainly developed in this sublibrary. The two sub-libraries can be used simultaneously through a structure-translation routine. Although "ZLIB" is developed mainly for supercomputers, the authors [2] have simultaneously tried to optimize the routines for scalar computers and therefore the use of "ZLIB" in scalar computers is also recommended.

** Operated by Universities Research Association, Inc., for the U.S. Department of Energy under the Contract No. DE-AC02-89ER40486.

0-7803-0135-8/91$01.00 ©IEEE

## II. The Algebra Of Truncated Power Series

In this section, the author is not trying to be mathematically rigorous. Once a variable, a function, or an operation is mentioned, its existence is assumed.

*(a) Symbolic convention*

Let $\vec{z}$ be an $n$-dimensional vector, i.e. its transpose can be expressed as

$$\vec{z}^T = [z_1, z_2, \ldots, z_n] ,$$

where $z_i$, for $i = 1, \ldots, n$, are scalar variables. For example, we can consider

$$\vec{z}^T = [z_1, z_2, \ldots, z_6] = [x, p_x, y, p_y, t, p_t]$$

as the transpose of a vector representing the 3-dimensional canononical coordinates and their conjugate momenta for an accelerator.

Let $U$ be a function of $\vec{z}$. This means $U$ is a function of $z_1, z_2, \ldots, z_n$. Its truncated power series (TPS) expansion up to an integer $\Omega$ order is expressed as

$$U(\vec{z}) = \sum_{k=0}^{\Omega} u(\vec{k}) \vec{z}^{\vec{k}} ,$$

where

$$\vec{z}^{\vec{k}} \equiv z_1^{k_1} z_2^{k_2} \ldots z_n^{k_n} ,$$

$$k = \sum_{i=1}^{n} k_i , \qquad \text{for } 0 \le k_i \le \Omega ,$$

$$\sum_{k=0}^{\Omega} \equiv \text{summation over all } \vec{k}\text{'s for } k = 0, 1, \ldots, \Omega .$$

Note that $U(\vec{z})$ is called an $n$-variable TPS, of order $\Omega$. The number of monomials for an $n$-variable TPS, of order $\Omega$, is given by

$$\eta = \frac{(n+\Omega)!}{n!\Omega!} .$$

A unit TPS is defined as

$$I(\vec{z}) = \sum_{k=0}^{\Omega} i(\vec{k}) \vec{z}^{\vec{k}} = 1 ;$$

i.e.

$$\begin{aligned} i(\vec{k}) &= 1 \qquad \text{for } k = 0 , \\ i(\vec{k}) &= 0 \qquad \text{for } k > 0 . \end{aligned}$$

Let $\vec{U}(\vec{z})$ be an $m$-dimensional vector TPS (VTPS), of $n$ variables, and of $\Omega$ order. It is expressed as

$$\vec{U}(\vec{z}) = \sum_{k=0}^{\Omega} \vec{u}(\vec{k})\vec{z}^{\vec{k}} ,$$

(i.e. $U_i(\vec{z}) = \sum_{k=0}^{\Omega} u_i(\vec{k})\vec{z}^{\vec{k}}$, for $i = 1, 2, \ldots, m$) where the transpose of $\vec{u}(\vec{k})$ is given by

$$\vec{u}^T(\vec{k}) = [u_1(\vec{k}), u_2(\vec{k}), \ldots, u_m(\vec{k})] .$$

One can consider $\vec{U}(\vec{z})$ as a map in accelerator physics.

A unit $n$-dimensional, $n$-variable VTPS of order $\Omega$, is defined as

$$\vec{I}(\vec{z}) = \sum_{k=0}^{\Omega} \vec{i}(\vec{k})\vec{z}^{\vec{k}} = \vec{z} .$$

Its transpose is given by

$$\vec{I}^T(\vec{z}) = \vec{z}^T = [z_1, z_2, \ldots, z_n] .$$

Numerically, the coefficients $u(\vec{k})$, $i(\vec{k})$, $\vec{u}(\vec{k})$, $\vec{i}(\vec{k})$ are used for representing $U(\vec{z})$, $I(\vec{z})$, $\vec{U}(\vec{z})$, $\vec{I}(\vec{z})$, respectively. Although $\vec{k}$ is a vector (multi-dimensional) index, due to memory limitation in computers, one-dimensional arrays are actually used for storing these coeffiecients

*(b) TPS Operations*

Addition:

$W(\vec{z}) = U(\vec{z}) + V(\vec{z}) \rightarrow w(\vec{k}) = u(\vec{k}) + v(\vec{k})$ for each $\vec{k}$.

Subtraction:

$W(\vec{z}) = U(\vec{z}) - V(\vec{z}) \rightarrow w(\vec{k}) = u(\vec{k}) - v(\vec{k})$ for each $\vec{k}$.

Multiplication:

$$W(\vec{z}) = U(\vec{z}) * V(\vec{z}) \rightarrow$$
$$w(\vec{j}) = \sum_{k=0}^{\Omega} u(\vec{k}) * v(\vec{j} - \vec{k}),$$

for each $(\vec{j} - \vec{k})_i \geq 0$, where $i = 1, 2, \ldots, n$.

Partial derivative:

$W(\vec{z}) = (\partial/\partial z_i)U(\vec{z}) \rightarrow w(\vec{j}) = (j_i + 1) * u(\vec{j} + \vec{1}_i)$,

where $i = 1, 2, \ldots,$ or $n$, and $\vec{1}_i$ is a unit vector in the $i^{\text{th}}$ dimension.

Partial integration:

$W(\vec{z}) = \int U(\vec{z})dz_i \rightarrow w(\vec{j}) = \left(\frac{1}{j_i}\right) * u(j - 1_i)$ for $j_i > 0$,

and, $w(\vec{j}) = 0$ for $j_i = 0$., where $i = 1, 2, \ldots,$ or $n$.

Using the above fundamental operations for the TPS, $w(\vec{j})$, the coefficients of $\vec{W}(\vec{z})$, can be obtained for the following basic TPS operations:

| | |
|---|---|
| Square: | $W(\vec{z}) = U^2(\vec{z})$, |
| Inversion: | $W(\vec{z}) = 1/U(\vec{z})$, |
| Division: | $W(\vec{z}) = U(\vec{z})/V(\vec{z})$, |
| Power: | $W(\vec{z}) = U^p(\vec{z})$, $p$ is an integer. |
| Square root: | $W(\vec{z}) = \text{sqrt}(U(\vec{z}))$, |
| Exponentiation: | $W(\vec{z}) = \exp(U(\vec{z}))$, |
| Logarithm: | $W(\vec{z}) = \ln(U(\vec{z}))$, |
| Trigonometry: | $W(\vec{z}) = \sin(U(\vec{z}))$, or |
| | $W(\vec{z}) = \cos(U(\vec{z}))$, |
| Poisson bracket: | $W(\vec{z}) = [U(\vec{z}), V(\vec{z})]$. |

*(c) VTPS Operations*

With the fundamental and the basic TPS operations ready, $\vec{w}(\vec{j})$, the coefficients of $\vec{W}(\vec{z})$, can be obtained for the following basic VTPS operations.

Concatenation: $\vec{W}(\vec{z}) = \vec{V}(\vec{U}(\vec{z}))$,

where, in the usual case, $\vec{U}$ is an $n$-dimensional $n$-variable VTPS, $\vec{V}$ and $\vec{W}$ are $m$-dimensional, $n$-variable VTPS, $m$ and $n$ may or may not be equal.

Inversion:

Given an $n$-dimensional, $n$-variable $\vec{U}(\vec{z})$, an $n$-dimensional, $n$-variable $\vec{U}^{-1}(\vec{z})$ can be obtained such that

$$\vec{U}^{-1}(\vec{U}(\vec{z})) = \vec{U}(\vec{U}^{-1}(\vec{z})) = \vec{I}(\vec{z}) .$$

All the above basic TPS or VTPS operations have been implemented in "ZLIB".

*(d) Tracking: $\vec{z}' = \vec{U}(\vec{z})$*

In conjunction with the implementation of the fundamental and basic TPS and VTPS operations, substitution of a numerical vector $\vec{z}$ into a VTPS (or a map) is implemented in the "ZLIB".

*(e) Dragt-Finn factorization:*

A closed-orbit truncated power series map can be expressed as a VTPS given by

$$\begin{aligned} \vec{z}' &= m(\vec{z}) : \vec{z} = \vec{U}(\vec{z}) = \sum_{k=1}^{\Omega} \vec{u}(\vec{k})\vec{z}^{\vec{k}} \\ &= M\vec{z} + \sum_{k=2}^{\Omega} \vec{u}(\vec{k})\vec{z}^{\vec{k}} \end{aligned}$$

Due to symplecticity, it can be converted into a Dragt-Finn factorization map given by [5]

$$\begin{aligned} \vec{z}' &= m(\vec{z}) : \vec{z} \\ &= \mathcal{A}^{-1}(\vec{z})\mathcal{R}(\vec{z})m_f(\vec{z})\mathcal{A}(\vec{z}) : \vec{z}, \end{aligned}$$

where $\mathcal{A}(\vec{z})$ and $\mathcal{R}(\vec{z})$ are the global forms of the normalized rotation R and its associated canonical generation matrix A such that $R = A^{-1}MA$ [6], and

$m_f(\vec{z}) = exp(: f_3(\vec{z}) :)exp(: f_4(\vec{z}) :) \cdots exp(: f_\Omega(\vec{z}) :)$,

where terms in each of the $k^{th}$ order TPS, $f_k(\vec{z})$, are $k^{th}$ order of $\vec{z}$ for $k = 3, 4, \cdots, \Omega$.

Dragt-Finn factorization of a truncated power series map and converting of a Dragt-Finn factorization map into a power series map together with some relevant Lie algebraic analysis of the map have been implemented in the "ZLIB".

## III. The ZLIB

"ZLIB" is a member of the Z-family programs which include (other than ZLIB): Zmap (a map extraction program), Ztrack (a vectorized and parallelized post-Teapot tracking program), Zremc1 and Zremc2 ($1\frac{2}{2}$ - and $2\frac{1}{2}$- dimensional relativistic electromagnetic particle simulation programs), and Zpcomp (a macro precompiler for fortran). Similar to the routines in the IMSL library which perform linear algebra through matrix operations, routines in "ZLIB" perform differential algebra through the operations of expanded power series, truncated at a pre-set order, to include nonlinear effects. Unlike linear algebra which has a domain idealized to be unlimited, differential algebra has a narrow domain where the power series converge at a reasonable rate, that is, the scope of differential algebra is restricted to problems for which an interest region (domain) can be identified to have a reasonable convergent rate for the power series expansion of the governing equations. Presently "ZLIB" finds its application in accelerator physics, since particles in an accelerator can only be stable in a region where the expanded power series of the nonlinear equations governing the system converge with a reasonable rate. Applications of "ZLIB" to other branches of physics, such as optics, should be possible.

Since "ZLIB" uses dynamic memory and includes most fundamental operations for differential algebra, a convenient way of using "ZLIB" would be to gather those binary object files of "ZLIB" subroutines into a library and load it with those commonly used on-line libraries such as fortlib, IMSL library, or graphics libraries and then keep this binary "ZLIB" on line. All that a user need to do is to load his program with ZLIB. Other libraries are automatically linked as long as they were loaded with ZLIB when ZLIB was created.

Before any subroutine using the data structure of the sublibrary "ZPLIB" is called, the user should include the following statement (assuming ZLIB 2.0 is used)

"call zpprep(nv,no,nm,npm),"

where "nv' and "no" are the number of variables and the maximum order the user desires; "nm", is a returned value for the number of monomials, i.e. nm=(nv+no)!/(nv!no!), is returned for the user; "npm" is the maximum number of particles. The user should set a small integer or 0 for npm if map tracking is not desired. once the statement "call zpprep(nv,no,nm,npm)" is executed, Data structure interger poniters and internal working memories are dynamically geneated. Occasionally, the user may wish to use routines in the sub-library "ZPLIB" to perform initialization (reading in a VTPS) and tracking only. In such a case, he may replace the statement "call zpprep(nv,no,nm,npm)," with the calling statement "call zptrkp(nv,no,nm,npm)," to save computer memory since less dynamical memory is generated in this case.

Once the statement, "call zpprep(nv,no,nm,npm)," is executed, all the TPS's are assumed to be nv-variable TPS's of order smaller than or equal to "no", and all the VTPS's are assumed to be nv-variable VTPS's of order "no", although operations can be performed up to orders that are lower than "no".

Similarly, to use the sublibrary "TPALIB", the preparation statement "call tpaprp(nv,no,nm,npm)" should be executed before any subroutine using the data structure of TPALIB is called. Note that slightly different from the ZPLIB, "no" is the order (not the maximum order) while "nv" is the maximum number of variables the user desires.

Once the statement "call tpaprp(nv,no,nm,npm)" is executed, all the TPS's and the VTPS are assumed to be order of "no," but not necessarily to be of nv variables. The number of variables can be smaller or equal to nv.

## IV. Future Expectation of ZLIB

Future direction for ZLIB development would be to develop routines that can weight the parameters differently from the canonical conjugate coordinates and momenta and analyze the parameterized map [7], An algorithm for fully parameterized Dragt-Finn factorization has been obtained [7]. With suitable modification of the existing technique [8], a parameterized nonlinear normal form of the map can also be programmed.

## V. Acknowledgments

Continuous support and encouragement for the development of this numerical library from Alex Chao is highly appreciated.

## VI. References

[1] M.Berz, in Proceedings of the 1989 IEEE Particle Accelerator Conference, 1989,

[2] Y. Yan and C. Yan, SSC Laboratory Report SSCL-300 (1990).

[3] Y. Yan, SSC Laboratory Report SSCL-299, 1990.

[4] L. Schachinger and R. Talman, Particle Accelerators, Vol. 22, 1987.

[5] A. J. Dragt and J. M. Finn, J. Math. Phys. 20, 2649 (1979)

[6] D. Edwards and L. Teng, in Proceedings of the 1973 IEEE Particle Accelerator Conference, p. 885 (1973); L.C. Teng, Fermi National Laboratory Report FN-229, 1971.

[7] Y. Yan, SSC Laboratory Report SSCL-460, 1991; Y. Yan SSCL-302, 1990.

[8] E. Forest, M. Berz, and J. Irwin, Particle Accel. 24, 91 (1989).