

A New Solution for Increasing Efficiency of Word-Lifting Algorithm

Zhong Zhou

College of Mathematics and Information Science, Zhongyuan University of Technology,
Zhengzhou 450007, China

southmonarch@qq.com

Abstract

It is well known that the intrinsic property of symbolic dynamics is coarse granulation, the inverse problem is to obtain the model parameters of a given periodic super stable kneading sequence (SSKS), it is the so-called word-lifting algorithm or technique. General, the word-lifting algorithm based on the newton iteration is quick and efficient, however, as the period length of a SSKS dramatic increase, more precision is need to reach a fixed point, it will spend more CPU time. In the paper, the word-lifting algorithm of symbolic dynamics for trimodal maps is considered and upgraded, by using of message passing interface (MPI) and cluster computing environment, the calculating tasks of nine partial derivatives in a Jacobian determinant of 3×3 are assigned to nine PCs on the same local net respectively, the time of the improved algorithm is reduced and efficiency increased by 3-6 times of the original.

Keywords

Symbolic dynamics; Word-lifting algorithm; MPI; Cluster; Jacobian determinant.

1. Introduction

Symbolic dynamics is an important tool and discipline for the study of dynamic system and chaos. Symbolization of a numerical orbit looks like a coarse granulation process and lost information, Kaplan [1] had proposed an algorithm to solve the inverse problem and soothed such concerns firstly, it was developed by B.-L. Hao in symbolic dynamics of unimodal maps later and named it the word-lifting algorithm or technique [2]. With the development of the 1D symbolic dynamics, the difficult of lacking explicit express in inverse function of polynomials to the 5th power or more leads to no way to research the symbolic dynamics in 1D m -modal maps if $m \geq 4$. We had proposed an effective numerical algorithm and solved this issue [3], in fact, the new word-lifting algorithm is quick and effective because of using Newton's iterative method of second-order convergence, the sole condition is that a proper initial point is given in the parameter space.

For a periodic SSKS W which pass through the m critical points, W determines a system of nonlinear equations. The word-lifting algorithm is to calculate the parameters of the trimodal maps corresponding W . Two factors may lead to sharp decline of algorithm speed, some times, the CPU time is too long to endure and lost practicality.

The first reason is the precision issue. It is well known the standard IEEE 754 delimits that double float precision could support about sixteen decimal significant digits under hardware, the CPU can obtain the best calculation speed by the aid of floating-point coprocessors. Once the sixteen decimal significant digits cannot meet the requirement of the calculation precision, the high precision library should be used to accomplish the computing tasks, more higher precision you set, more slower speed you will obtain. For example, two SSKSs $W_1 = RL^{99}C$ and

$W_2 = RL^{100}C$ in unimodal maps, μ_1 and μ_2 are parameters of W_1 and W_2 in iterative maps $x_{n+1} = 1 - \mu x_n^2$ respectively, in order to distinguish them, forty decimal significant digits are needed at least during the calculation. Another case will arouse the more precision requirement, it is the error accumulation. With the length of W^{*n} in the $|W|$ -tupling bifurcation leaping, every calculation of function includes many composite inverse limbs maps which will enlarge the error accumulations.

The second reason comes from the non-normal star products which may be not corresponding a contraction mapping, it will lead to the demand for the super high precision. The non-normal star products are found firstly in trimodal maps and may produce a route to chaos by W^{*n} in $|W|$ -tupling bifurcation process [4]. The generalized Feigenbaum constants is worthy of probing and researching.

The two reasons decide the infinite precision libraries should be used inevitably. The infinite or high precision libraries are always supported for open source, for example, the famous Gmp, Ntl in C++ or mpmath in Python and so on. However, the using of the libraries must lead to a lower speed of calculating.

In this paper, we found a solution to enhance the calculating speed. The word-lifting algorithm in trimodal maps will compute a Jacobian determinant of 3×3 , including calculation of nine partial derivatives, it consists of the main part CPU time. Because of property of the iteration, the $n+1$ iteration needs the result of n th iteration, the newton-iteration does not look suitable for parallel algorithm and architecture, the MPI is fit to accomplish the task [5]. Every calculation of nine partial derivatives will be assigned to one of nine PCs on the same local net, by the use of instructions of MPI, cooperation and waiting, one of the nine PCs may act as the host which is in charge of broadcasting and waiting receiving the messages from other nodes, the host pc accomplishes the newton-iteration in the word-lifting algorithm. Finally, the calculation speed holds 3-6 times than that of unimproved, the net speed and CPU waiting time leads that theoretically 9 times promotion could not reach actually.

The paper is organized as follows. In sec.2, the word-lifting algorithm is introduced for SSKS of symbolic dynamics of trimodal maps; in sec.3, the MPI cluster configuration and application in word-lifting algorithm ; in sec.4, conclusion.

2. Word-Lifting Algorithm of Symbolic Dynamics of Trimodal Maps

2.1. The Iteration Modal of Trimodal Maps

Consider the general trimodal maps $f(x) = k \int (x-c)(x-d)(x-e)dx + b$, on the interval $[-1,1]$ of endomorphism, by the two boundary conditions $f(-1)=-1$ and $f(1)=-1$, parameters k and b are eliminated. In fact, The The iterated map of trimodal is written as

$$x_{n+1} = f(x_n, c, d, e) \quad (1)$$

Here, c, d and e are horizontal coordinates of three critical points C, D and E , while L, M, N and R are four monotonous limbs, by the MSS order [6], $L \prec C \prec M \prec D \prec N \prec E \prec R$ holds. For an initial point x_0 , by iterative map (1), a numerical sequence is obtained as x_0, x_1, \dots, x_n , it can be converted into a symbolic sequence $Q_0 Q_1 \dots Q_n \dots$, by the following rule (2), the coarse granulation process is quintessence of the symbolic dynamics.

$$Q_k = \begin{cases} L & \text{if } x_k < c, \\ C & \text{if } x_k = c, \\ M & \text{if } c < x_k < d, \\ D & \text{if } x_k = d, \\ N & \text{if } d < x_k < e, \\ E & \text{if } x_k = e, \\ R & \text{if } x_k > e. \end{cases} \quad k \in \mathbb{Z}^+ \quad (2)$$

If the sequence is periodic, for example $(ZEXDYC)^\infty$ or $(XDZEYC)^\infty$, it is simply normalized as $ZEXDYC$ or $XDZEYC$, where Z, X, Y are sequences composed of $\{L, M, N, R\}$, these sequences are called TSSKS which are periodic and passed through the three critical points C, D and E respectively. If the number of critical points is unstressed, TSSKS is called SSKS. The rule of sort between two sequences and the admissible conditions are omitted here [7].

2.2. The Word-Lifting Algorithm of TSSKS

A TSSKS is important to calculate its corresponding parameters c, d and e . Where K_E, K_D and K_C are subsequences of E, D and C respectively, while a system of three nonlinear equations are decided by the TSSKS while its parameters must exist and be sole because of the admissibility, they can be calculated by Newton iteration algorithm if a proper initial value is given. By eq. (1), The digital orbits of TSSKSs $ERE DC$ and $RLDLE C$ are drawn (Fig.1).

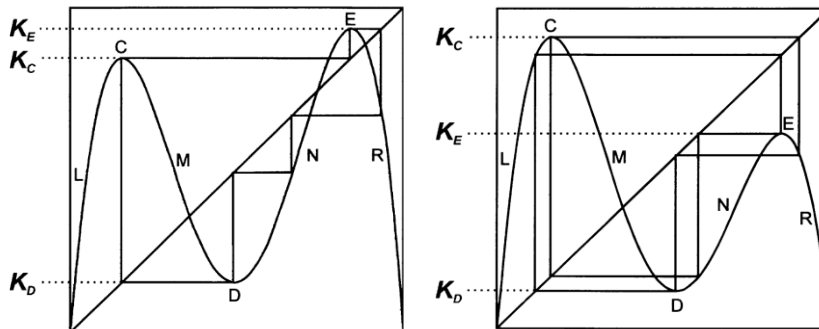


Fig. 1 Schematic iterative map for TSSKSs $ERE DC$ (left) and $RLDLE C$ (right)

Here, we presented a flow chart of the word-lifting algorithm for 1D trimodal maps in Fig.2.

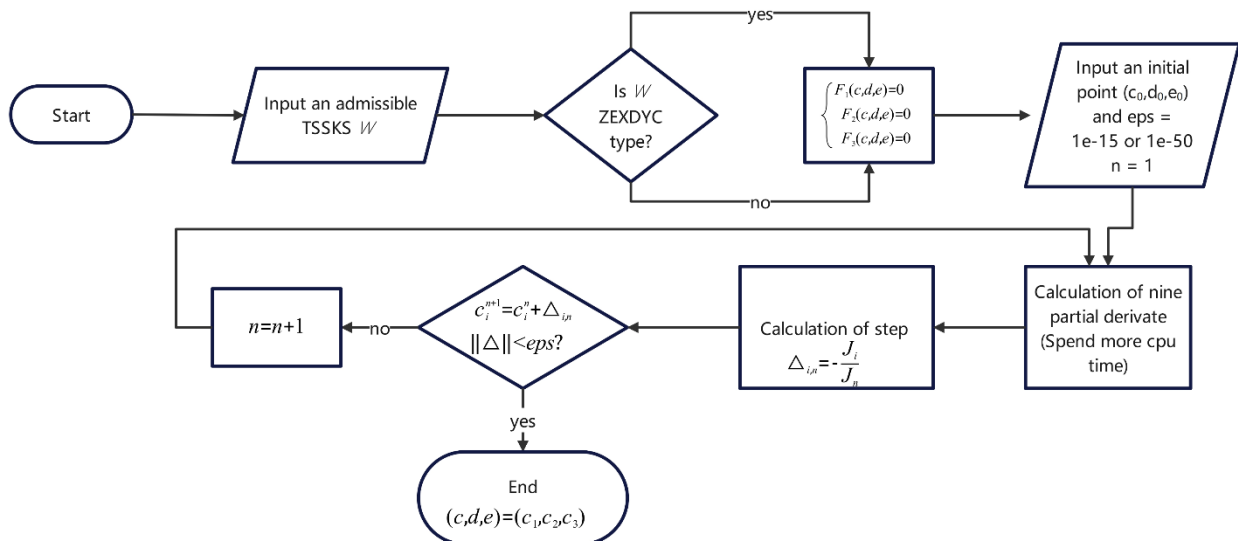


Fig.2 Flow chart of the word-lifting algorithm for 1D trimodal maps

Now, we present a concrete example of TSSKS *RLDLEC* to show the whole process of word-lifting algorithm. First, *RLDLEC* is *XDZEYC* type, according to the order of three critical points, $RLDLEC = (RLDLEC)^\infty = CRLDLEC \dots$, three equations are obtained as

$$\begin{cases} f(c, c, d, e) = f_R^{-1} \circ f_L^{-1}(d, c, d, e) \\ f(d, c, d, e) = f_L^{-1}(e, c, d, e) \\ f(e, c, d, e) = c \end{cases} \tag{3}$$

We have equivalently

$$\begin{cases} F_1(c, d, e) = f(c, c, d, e) - f_R^{-1} \circ f_L^{-1}(d, c, d, e) = 0 \\ F_2(c, d, e) = f(d, c, d, e) - f_L^{-1}(e, c, d, e) = 0 \\ F_3(c, d, e) = f(e, c, d, e) - c = 0 \end{cases} \tag{4}$$

Where, f is (1), $f_L^{-1}, f_M^{-1}, f_N^{-1}$ and f_R^{-1} are four inverse limbs functions which may be accomplished by numerical method [3]. Different type of TSSKS decides the construction method of equations (4). For an admissible TSSKS W , a system of nonlinear equations like (4) is uniquely determined and may be programmed.

Second, a initial point $(c_0, d_0, e_0) = (-0.685, 0.083, 0.726)$ is given empirically, ϵ takes $1e-15$ or $1e-50$ for the different precision need, the former should set data type of the floating variables as double, the later should be RR which is from the famous NTL in C++ with high-precision, `#include<NTL/RR.h>` is placed on the front of your C++ codes, `RR::SetOutputPrecision(50)` and `RR::SetPrecision(50)` may set the decimal output precision and actual decimal precision, while set the iteration count $n = 1$.

Third, the calculation of iteration steps. Jacobian determinant is

$$J_n = \frac{\partial(F_1, F_2, F_3)}{\partial(c, d, e)} \Big|_{P_n} = \begin{vmatrix} \frac{\partial F_1}{\partial c} & \frac{\partial F_1}{\partial d} & \frac{\partial F_1}{\partial e} \\ \frac{\partial F_2}{\partial c} & \frac{\partial F_2}{\partial d} & \frac{\partial F_2}{\partial e} \\ \frac{\partial F_3}{\partial c} & \frac{\partial F_3}{\partial d} & \frac{\partial F_3}{\partial e} \end{vmatrix}_{P_n} \tag{5}$$

Where, $J_n^i = J_n \Big|_{ith \text{ column} \rightarrow (F_1, F_2, F_3)}$, iteration step $\Delta_{i,n} = -\frac{J_n^i}{J_n}, i = 1, 2, 3.$, the newton iteration is as follows,

$$\begin{cases} c_{n+1} = c_n + \Delta_{1,n}, \\ d_{n+1} = d_n + \Delta_{2,n}, \\ e_{n+1} = e_n + \Delta_{3,n}. \end{cases} \tag{6}$$

For example, one of the nine partial derivatives is $\frac{\partial(F_1)}{\partial c} = \frac{F_1(c+h, d, e) - F_1(c-h, d, e)}{2h}$, here h takes $\epsilon * 1e-3$ for the cause of error accumulations. It is worthy of noting that another algorithm of $\frac{\partial(F_1)}{\partial c}$ is $\frac{F_1(c+h, d, e) - F_1(c, d, e)}{h}$ and seems the counts of calculation of F_i is less than that of the former during the whole calculation of J_n and J_n^i , however, the former will get more significant digits in an iteration process, overall the CPU time is spared and more efficient.

Fourth, if the norm of $\Delta_{i,n}$ is less than eps, the calculation of newton iteration would reach the fixed point $(c_{n+1}, d_{n+1}, e_{n+1})$ and terminate, else continue the next iteration. Generally, a numerical orbit is drawn by the iteration map (1) like right part of Fig. 2., it is a corroboration of the validity of the word-lifting algorithm.

Remark, the fixed point may be not reached sometimes, it comes from the unfit initial point, in fact the admissibility of TSSKS W decides the existence and uniqueness of the fixed point or solution of the system of three nonlinear equations, if it happens, we would try any other initial point to find the solution, on the other hand, the selection of a higher precision may solve the problem quickly.

3. Improvement of Word-Lifting Algorithm Based on the MPI Cluster

3.1. Construction of the MPI Cluster Environment

MPI is the abbreviation of message process interface, it develops to a parallel programming language which supports C++, Python and so on. MPI support two kinds of modes, the first is the so called MP (multi-processor), it supports MPs in one computer and makes full use of every CPU kernel, the second mode supports every computer on the same local net, these computers form a cluster, the so called MPI cluster computing environment. In this paper, the MPI cluster mode is selected for improvement of word-lifting algorithm. The reasons are as follows. First, the MP mode needs computer has ten and above CPU kernels, early computers in campus seldom meet the requirement, on the other hand, for a computer with ten CPU kernels runs the calculating codes, the computer CPU utilization reaches 90%, however, the total improvement has almost 2-3times because of the wait time amount the thread processes, the computer could do nothing meanwhile. Second, the MPI cluster mode has low hardware requirements, a group of PCs with same configuration and on the same local net is easily found in a campus computer room, it is less expensive. See Fig.3 .

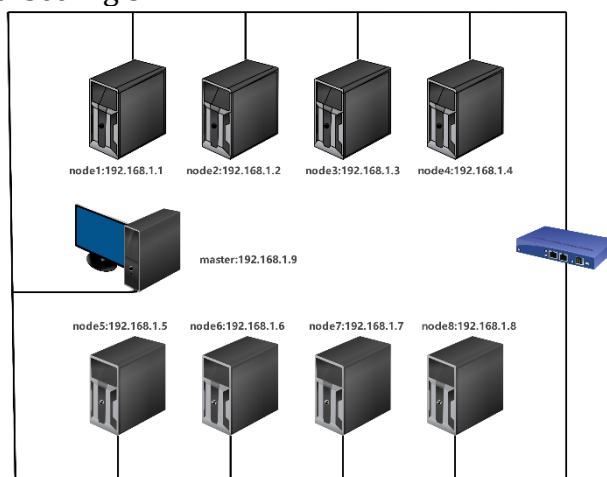


Fig.3 Nine calculation nodes in a local ethernet

MPI cluster configuration supports all sorts of operation system, such as Unix, Macintosh, Windows and so on. Here, we select the Windows 10 64 bit operation system and visual studio 2019. First, we go to official website to download the MPI SDK mpich2 package of windows version [8]. Install the mpich2 package in the nine PCs of the same local net. One of nine computers is named master, other computer names node1,node2,...,node8 respectively, these nine names are stored in a text file named nodenames.txt, including static ip address of every computer node. These names of computer are used to undertake or allocate the tasks of nine partial derivatives. Mpiexec is an important tool to set options for the parallel codes starter. For example, on the rear of command line prompt, we type the following code and type enter key.

mpixec -hostfile nodenames.txt -np 9 -npsocket 1 d:\word-lift-mpi\wlm.exe (7)

It will set nine computer nodes on the local net and every node has one process, meanwhile start the main program wlm.exe in every computer node, here wlm.exe is static and obtained in advance by compiling source code file wlm.cpp and link all static library in VS2019 with MPI environment. If command (7) may start the program at once, the MPI cluster environment is configured successfully.

3.2. Word-Lifting Algorithm Based on the MPI Cluster

In the front of the main program, the calculation environment of MPI cluster should be initialized with MPI_Init(&argc, &argv), while MPI_Finalize() in the end. The main MPI instructions is as follows:

```
#include<mpi.h>
MPI_Comm_size()
MPI_Comm_rank()
MPI_Comm_rank()
MPI_Get_processor_name()
MPI_Send()
MPI_Recv()
```

The simple flow chart of word-lifting algorithm on MPI cluster for 1D trimodal maps is seen Fig. 4.

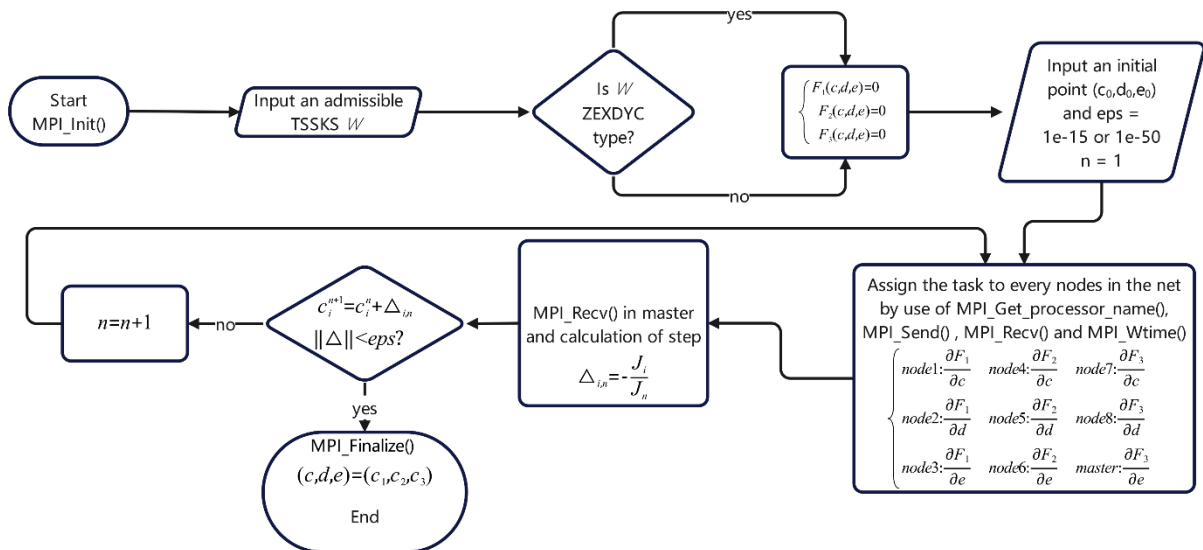


Fig. 4 Flow chart of the word-lifting algorithm on MPI cluster for 1D trimodal maps

All actual C++ codes about fig.4 are omitted here, it is the first attempt to improve the efficiency of iterative system by MPI cluster calculation. Here, we have obtained routes to chaos by the star product of trimodal maps, such as $(EDC)^*n$, $(EMDC)^*n$ and $(RERDC)^*n$, in Tab.1 CPU time1 is using the general word-lifting algorithm, while CPU time2 using the that of the MPI cluster calculation, there are about 3-6 times improvement apparently, the result is quite satisfactory.

Table 1 Comparison of CPU time before and after improvement of the algorithm

W^{*n}	Period	Parameters c,d,e	CPU time1 seconds	CPU time2 seconds	Improvement rate
$(EDC)^{*7}$	2187	-0.667065922633715449 0.087499633213138016 0.702593222756168563	72.653	21.671	335%

$(EMDC)^{*7}$	16384	-0.647135841231475018 0.191366420114692312 0.725193437829363331	387.458	89.234	434%
$(RERDC)^{*6}$	46656	-0.695379721169717940 -0.011796802762307013 0.690191078679068977	1215.924	215.207	565%

4. Conclusion

The result in Tab.1 shows that the improvement algorithm in MPI cluster calculation environment for word-lifting algorithm is quite efficient. It provides an idea to improve efficiency of parallel computing for iterative system. However, if the word-lifting algorithm for 1D m -modal maps, it will need m^2 computers in the same local ethernet, the improvement rate would be researched furtherly.

References

- [1] Kaplan H.: New method for calculating stable and unstable periodic orbits of one-dimensional maps, Phys. Lett. A, Vol. 97(1983) p365.
- [2] B.-L. Hao: Elementary Symbolic Dynamics and Chaos in Dissipative Systems (World Scientific, Singapore, 1989).
- [3] Z.Zhou,K.F. Cao: An effective numerical method of the word-lifting technique in one-dimensional multimodal maps, Phys. Lett. A, Vol. 310(2003) No.1, p52-59.
- [4] Zhou Z, Peng S-L: Cyclic star products and universalities in symbolic dynamics of trimodal maps. Physica D, Vol. 140(2000), 213-226..
- [5] Michael J. Quinn: MPI and Open MP Parallel Programming Language(Tsinghua University Press, Beijing, 2004).
- [6] Metropolis N, Stein M L, Stein P R: On finite limit sets for transformations on the unit interval. J Comb Theory A, Vol. 15(1973), 25-44.
- [7] Zhou Z, Peng S.-L.: Star Products of Quartic Maps with Two Equally High Peaks, Chinese Physics Letter, Vol. 17(2000), 252~254.
- [8] <https://www.mpich.org/downloads/>.