A Maple package for verifying ultradiscrete soliton solutions

Min Gao [a], Masaaki Ito [b,*]

[a] *Department of Computer Science, Minjiang University, Fuzhou Fujian 350108, China*

[b] *Department of Applied Mathematics, Faculty of Engineering, Hiroshima University, Higashi-Hiroshima 739-8527, Japan*

Abstract

We present a computer algebra program for verifying soliton solutions of ultradiscrete equations in which both dependent and independent variables take discrete values. The package is applicable to equations and solutions that include the max function. The program is implemented using *Maple* software.

Key words: Ultradiscrete equations, Soliton solutions, Max-plus algebra, Computer algebra
PACS: 02.30.Ik, 02.70.Wz

Program summary

Title of program : Ultde
Catalogue identifier :
Program obtainable from : CPC Program Library, Queen's University of Belfast, N. Ireland
Computers : PC/AT compatible machine
Operating systems : Windows 2000, Windows XP
Programming language used : Maple 10
Memory required to execute with typical data: Depends on the problem; minimum about 1G bytes.
No. of bits in a word : 32
No. of bytes in distributed program, including test data, etc. : 3904
Restriction on the complexity of the problem : The program can only handle

* Corresponding author.
Email address: ito@amath.hiroshima-u.ac.jp (Masaaki Ito).

single ultradiscrete equations.
Typical running time : Depends on the complexity of the equation and solution.

1 Introduction

Integrable systems comprise a primary subjects in mathematics and physics. Soliton equations in particular have been studied as integrable systems. In recent years, the ultradiscrete version of soliton equations has attracted a great deal of attention [1-4]. Ultradiscretization is a procedure transforming a discrete equation into an ultradiscrete equation in which both dependent and independent variables take discrete values [1,5]. The formula for ultradiscretization is given by

$$\lim_{\varepsilon \to 0} \varepsilon \log(e^{A/\varepsilon} + e^{B/\varepsilon}) = \max(A, B) \tag{1}$$

for arbitrary real numbers A and B. Using this formula, the field of real numbers can be transformed into a so-called "max-plus" algebra. Moreover, this formula has been used to transform various soliton equations to cellular automata without losing the mathematical properties of the corresponding difference equation. However, solutions of ultradiscrete equation are not necessarily derived from those of the corresponding discrete equation. Therefore, it is important to find solutions for ultradiscrete equations directly. In ultradiscrete systems, evolution equations and/or solutions often include the max function. In this case, to verify a solution of an ultradiscrete equation, we must divide the domain of parameters included in the solution into appropriate intervals and execute the max function in all cases. If the resulting expression includes many parameters as in multi-soliton solutions, enormous calculations are required to verify the solution. The use of computer algebra can be helpful in such calculations. Computer algebra systems, such as *Maple* and *Mathematica*, already have the mathematical capability to execute the max function. *Mathematica* is particularly apt for simplification calculations, and it can thus verify very simple solutions. However, for expressions with many parameters, the built-in max function cannot complete verification.

In this paper, we propose an original algorithm for verifying soliton solutions of ultradiscrete equations, and implement the algorithm using *Maple*. This paper is organized as follows. In Section 2, we briefly review the ultradiscretization of discrete soliton equations and resulting solutions. In Section 3, we outline the algorithm for verifying ultradiscrete soliton solutions. In Section 4, we give a description of the *Maple* package `Ultde`. In Section 5, we present some examples to illustrate how to verify ultradiscrete soliton solutions. A conclusion is given in Section 6.

2 Ultradiscretization of soliton equation and its solutions

As an example, we take the discrete Lotka-Volterra equation proposed by Hirota and Tsujimoto [6],

$$\frac{1}{\delta}(u_n^{m+1} - u_n^m) = u_n^m u_{n-1}^m - u_n^{m+1} u_{n+1}^{m+1}. \tag{2}$$

This equation possesses N-soliton solutions and infinite conserved quantities. We transform the variables and constants using the following functions, which include a parameter ε:

$$u_n^m = e^{U_n^m/\varepsilon},\ \delta = e^{-1/\varepsilon}. \tag{3}$$

Taking the limit $\varepsilon \to +0$, we obtain an ultradiscrete Lotka-Volterra equation [7],

$$U_n^{m+1} - U_n^m = \max(0, U_{n-1}^m - 1) - \max(0, U_{n+1}^{m+1} - 1). \tag{4}$$

If initial values of U are integers, any U is also an integer. The procedure used to generate a completely discrete equation like (4) from a difference one like (2) using the transformation and limit as stated above is called "ultradiscretization."

By using the transformation of U to F,

$$U_n^m = F_{n-1}^m + F_{n+2}^{m+1} - F_n^m - F_{n+1}^{m+1}, \tag{5}$$

we obtain another form of the ultradiscrete Lotka-Volterra equation,

$$\begin{aligned} &F_{n-1}^{m+1} + F_{n+2}^{m+2} + F_n^m + F_{n+1}^{m+1} + \max(0, F_n^{m+1} + F_{n+3}^{m+2} - F_{n+1}^{m+1} - F_{n+2}^{m+2} - 1) \\ &= F_{n-1}^m + F_{n+2}^{m+1} + F_n^{m+1} + F_{n+1}^{m+2} + \max(0, F_{n-2}^m + F_{n+1}^{m+1} - F_{n-1}^m - F_n^{m+1} - 1). \end{aligned} \tag{6}$$

An ultradiscrete multi-soliton solution of (6) is derived from that of (2) through the same ultradiscretizing procedure. One-, two- and three-soliton solutions are expressed as follows:
One-soliton solution:

$$\begin{cases} F_n^m = \max(0, \Xi_1), \\ \Xi_1 = K_1 n - \Omega_1 m + \Xi_1^0, \\ \Omega_1 = \max(0, K_1 - 1) - \max(0, -K_1 - 1). \end{cases} \tag{7}$$

Two-soliton solution:

$$\begin{cases} F_n^m = \max(0, \Xi_1, \Xi_2, \Xi_1 + \Xi_2 + A_{12}), \\ \Xi_i = K_i n - \Omega_i m + \Xi_i^0, \\ \Omega_i = \max(0, K_i - 1) - \max(0, -K_i - 1), \quad i = 1, 2, \\ A_{12} = |K_1 - K_2| - |K_1 + K_2|. \end{cases} \tag{8}$$

Three-soliton solution:

$$\begin{cases} F_n^m = \max(0, \Xi_1, \Xi_2, \Xi_3, \Xi_1 + \Xi_2 + A_{12}, \Xi_1 + \Xi_3 + A_{13}, \\ \qquad \Xi_2 + \Xi_3 + A_{23}, \Xi_1 + \Xi_2 + \Xi_3 + A_{12} + A_{13} + A_{23}), \\ \Xi_i = K_i n - \Omega_i m + \Xi_i^0, \\ \Omega_i = \max(0, K_i - 1) - \max(0, -K_i - 1), \quad i = 1, 2, 3, \\ A_{ij} = |K_i - K_j| - |K_i + K_j|, \quad i \neq j, \ i, j = 1, 2, 3. \end{cases} \tag{9}$$

We can verify one- and two-soliton solutions directly by hand, but in the case of three- or multi-soliton solutions, an extremely tedious and enormous calculation is required. To this end, we propose an algorithm for verifying higher-order soliton solutions.

3 Algorithm for verifying ultradiscrete soliton solutions

In our algorithm, we consider the following max-plus equation:

$$L_1(F_n^m, F_n^{m\pm1}, \ldots, F_{n\pm1}^m, F_{n\pm1}^{m\pm1}, \ldots) = L_2(F_n^m, F_n^{m\pm1}, \ldots, F_{n\pm1}^m, F_{n\pm1}^{m\pm1}, \ldots), \tag{10}$$

where L_1 and L_2 are linear combinations of $F_{n\pm i}^{m\pm j}$ $(i, j = 0, \pm1, \pm2, \ldots)$, which includes the max function. The N-soliton solution of Eq. (10) is assumed to have the following form:

$$\begin{cases} F_n^m = \max\limits_{\mu_i=0,1} \left(\sum\limits_{i=1}^{N} \mu_i \Xi_i(m, n) + \sum\limits_{1 \leq i < j \leq N} \mu_i \mu_j A_{ij} \right), \\ \Xi_i(m, n) = K_i n - \Omega_i m + \Xi_i^0, \\ \Omega_i = \Omega_i(K_i), \quad i = 1, 2, \ldots, N, \\ A_{ij} = A_{ij}(K_i, K_j), \quad i \neq j, \ i, j = 1, 2, \ldots, N, \end{cases} \tag{11}$$

where $\max_{\mu_i=0,1}$ denotes the maximum value over all possible combinations of $\mu_1 = 0, 1$, $\mu_2 = 0, 1$, $\cdots$, $\mu_N = 0, 1$, and Ω_i, A_{ij} include the max and/or absolute value functions. The algorithm has three steps.

Step 1. *Convert each side of equation into a single-max expression*

Substituting the solution (11) into (10), we have an expression in which the max function is deeply nested. To convert the expression into a single-max function form, we recursively use the following properties of max-plus algebra:

$$\begin{cases} \max(A, B) + C = \max(A + C, B + C), \\ \max(A, \max(B, C)) = \max(A, B, C). \end{cases} \tag{12}$$

The above step can be performed by using the simplification function in *Mathematica*. However, for expressions with many parameters, such as multi-soliton solutions, the simplification cannot be completed. In *Maple*, such a simplification likewise cannot be implemented.

For example, the expression $\max(a_1, a_2) + \max(b_1, b_2) + c$ is converted into a single-max expression as follows:

$$\begin{aligned} &\max(a_1, a_2) + \max(b_1, b_2) + c \\ &= \max(a_1 + \max(b_1, b_2), a_2 + \max(b_1, b_2)) + c \\ &= \max(\max(a_1 + b_1, a_1 + b_2), \max(a_2 + b_1, a_2 + b_2)) + c \\ &= \max(a_1 + b_1, a_1 + b_2, a_2 + b_1, a_2 + b_2) + c \\ &= \max(a_1 + b_1 + c, a_1 + b_2 + c, a_2 + b_1 + c, a_2 + b_2 + c). \end{aligned}$$

Setting $x_i = K_i n - \Omega_i m + \Xi_i^0$ $(i = 1, 2, \ldots, N)$, the N-soliton solution (11) is expressed as

$$F_{n+a}^{m+b} = \max_{\mu_i=0,1} \Big(\sum_{i=1}^{N} \mu_i (x_i + K_i a - \Omega_i b) + \sum_{1 \le i < j \le N} \mu_i \mu_j A_{ij}\Big). \tag{13}$$

Substituting the soliton solution (13) into the max equation (10) leads to a max-plus equation in the following form:

$$\max(l_1, l_2, l_3, \ldots) = \max(l_1', l_2', l_3', \ldots), \tag{14}$$

where

$$\begin{aligned} &l_i = \phi_i + h_i, \quad \phi_i = \sum_{j=1}^{N} a_{ij} x_j, \\ &h_i = \sum_{j=1}^{N} (b_{ij} K_j + c_{ij} \Omega_j + \sum_{l=j+1}^{N} d_{ij,l} A_{jl}) + e_i, \end{aligned} \tag{15}$$

and

$$
\begin{aligned}
&l'_i = \phi'_i + h'_i, \quad \phi'_i = \sum_{j=1}^{N} a'_{ij} x_j, \\
&h'_i = \sum_{j=1}^{N} (b'_{ij} K_j + c'_{ij} \Omega_j + \sum_{l=j+1}^{N} d'_{ij,l} A_{jl}) + e'_i.
\end{aligned}
\tag{16}
$$

Here $a_{ij}, b_{ij}, c_{ij}, d_{ij,l}, e_i, a'_{ij}, b'_{ij}, c'_{ij}, d'_{ij,l}, e'_i$ are constants, and ϕ_i, ϕ'_i correspond to phase terms in a continuous or discrete system, respectively. Note that we suppress the replacement of variables Ω_i and A_{ij} by the expressions $\Omega_i(K_i)$ and $A_{ij}(K_i, K_j)$ in this step.

As an illustration of our algorithm, we consider the ultradiscrete Lotka-Volterra equation (6). By adding 1 to each side of (6) and using the properties listed above (12), we obtain the following form:

$$
\begin{aligned}
&\max(F_{n-1}^{m+1} + F_{n+2}^{m+2} + F_n^m + F_{n+1}^{m+1} + 1, F_n^{m+1} + F_{n+3}^{m+2} + F_{n-1}^{m+1} + F_n^m) \\
&= \max(F_{n-1}^m + F_{n+2}^{m+1} + F_n^{m+1} + F_{n+1}^{m+2} + 1, F_{n-2}^m + F_{n+1}^{m+1} + F_{n+1}^{m+2} + F_{n+2}^{m+1}).
\end{aligned}
\tag{17}
$$

Setting $x_1 = K_1 n - \Omega_1 m + \Xi_1^0$, the one-soliton solution (7) is expressed as

$$
F_{n+a}^{m+b} = \max(0, x_1 + K_1 a - \Omega_1 b). \tag{18}
$$

Substituting (18) into (17) and using the properties listed above (12), the left-hand side of Eq. (17) leads to

$$
\begin{aligned}
&\max(0, 1, x_1, x_1 + 1, x_1 - \Omega_1, x_1 - K_1 - \Omega_1, x_1 - K_1 - \Omega_1 + 1, x_1 + K_1 - \Omega_1 + 1, \\
&\quad x_1 + 2K_1 - 2\Omega_1 + 1, x_1 + 3K_1 - 2\Omega_1, 2x_1 - 2\Omega_1 + 1, 2x_1 - \Omega_1, 2x_1 - K_1 - 2\Omega_1, \\
&\quad 2x_1 - K_1 - \Omega_1, 2x_1 - K_1 - \Omega_1 + 1, 2x_1 + K_1 - 3\Omega_1 + 1, 2x_1 + K_1 - \Omega_1 + 1, \\
&\quad 2x_1 + 2K_1 - 3\Omega_1, 2x_1 + 2K_1 - 2\Omega_1 + 1, 2x_1 + 3K_1 - 3\Omega_1, 2x_1 + 3K_1 - 3\Omega_1 + 1, \\
&\quad 2x_1 + 3K_1 - 2\Omega_1, 3x_1 - 2\Omega_1 + 1, 3x_1 - K_1 - 2\Omega_1, 3x_1 + K_1 - 3\Omega_1 + 1, \\
&\quad 3x_1 + 2K_1 - 4\Omega_1, 3x_1 + 2K_1 - 4\Omega_1 + 1, 3x_1 + 2K_1 - 3\Omega_1, 3x_1 + 3K_1 - 3\Omega_1, \\
&\quad 3x_1 + 3K_1 - 3\Omega_1 + 1, 4x_1 + 2K_1 - 4\Omega_1, 4x_1 + 2K_1 - 4\Omega_1 + 1),
\end{aligned}
\tag{19}
$$

and the right-hand side of the equation leads to

$$
\begin{aligned}
&\max(0, 1, x_1 - \Omega_1 + 1, x_1 - 2K_1, x_1 - K_1 + 1, x_1 + K_1 - 2\Omega_1, x_1 + K_1 - 2\Omega_1 + 1, \\
&\quad x_1 + K_1 - \Omega_1, x_1 + 2K_1 - \Omega_1, x_1 + 2K_1 - \Omega_1 + 1, 2x_1 - 2\Omega_1 + 1, 2x_1 - \Omega_1, \\
&\quad 2x_1 - K_1 - 2\Omega_1, 2x_1 - K_1 - \Omega_1, 2x_1 - K_1 - \Omega_1 + 1, 2x_1 + K_1 - 3\Omega_1 + 1, \\
&\quad 2x_1 + K_1 - \Omega_1 + 1, 2x_1 + 2K_1 - 3\Omega_1, 2x_1 + 2K_1 - 2\Omega_1 + 1, 2x_1 + 3K_1 - 3\Omega_1, \\
&\quad 2x_1 + 3K_1 - 3\Omega_1 + 1, 2x_1 + 3K_1 - 2\Omega_1, 3x_1 - 3\Omega_1, 3x_1 - 3\Omega_1 + 1, 3x_1 + K_1 - 3\Omega_1, \\
&\quad 3x_1 + K_1 - 2\Omega_1, 3x_1 + K_1 - 2\Omega_1 + 1, 3x_1 + 2K_1 - 3\Omega_1 + 1, 3x_1 + 3K_1 - 4\Omega_1 + 1, \\
&\quad 3x_1 + 4K_1 - 4\Omega_1, 4x_1 + 2K_1 - 4\Omega_1, 4x_1 + 2K_1 - 4\Omega_1 + 1).
\end{aligned}
\tag{20}
$$

Step 2. *Collect the elements of the same phase and generate equations that must be satisfied*

If the function (11) is a solution of (10), equation (14) must be satisfied for any value of x_i, that is, for any value of phase ϕ. As seen in Step 1, in general, there are many elements of the same phase in (14). Hence, we collect the elements of phase ϕ and reduce them to one element.

For example, if a max function includes two distinct phase Φ_1, Φ_2, and an element l_i is given by

$$l_i = \begin{cases} \Phi_1 + h_i, & i = 1, 2, \ldots, r, \\ \Phi_2 + h_i, & i = r+1, \ldots, s, \end{cases}$$

then $\max(l_1, l_2, \ldots, l_r, l_{r+1}, \ldots, l_s)$ is reduced as follows:

$$\begin{aligned} &\max(l_1, l_2, \ldots, l_r, l_{r+1}, \ldots, l_s) \\ &= \max(\Phi_1 + h_1, \Phi_1 + h_2, \ldots, \Phi_1 + h_r, \Phi_2 + h_{r+1}, \ldots, \Phi_2 + h_s) \\ &= \max(\Phi_1 + \max(h_1, h_2, \ldots, h_r), \Phi_2 + \max(h_{r+1}, \ldots, h_s)). \end{aligned}$$

Using the above procedure, (14) leads to

$$\begin{aligned} &\max(\Phi_1 + \max(h_1^{(1)}, h_1^{(2)}, \ldots, h_1^{(p_1)}), \Phi_2 + \max(h_2^{(1)}, h_2^{(2)}, \ldots, h_2^{(p_2)}), \ldots, \\ &\quad \Phi_M + \max(h_M^{(1)}, h_M^{(2)}, \ldots, h_M^{(p_M)})) \\ &= \max(\Phi_1 + \max(h_1'^{(1)}, h_1'^{(2)}, \ldots, h_1'^{(q_1)}), \Phi_2 + \max(h_2'^{(1)}, h_2'^{(2)}, \ldots, h_2'^{(q_2)}), \ldots, \\ &\quad \Phi_M + \max(h_M'^{(1)}, h_M'^{(2)}, \ldots, h_M'^{(q_M)})), \end{aligned} \tag{21}$$

where

$$\begin{aligned} h_i^{(k)} &= \sum_{j=1}^{N} (b_{ij}^{(k)} K_j + c_{ij}^{(k)} \Omega_j + \sum_{l=j+1}^{N} d_{ij,l}^{(k)} A_{jl}) + e_i^{(k)}, \\ h_i'^{(k)} &= \sum_{j=1}^{N} (b_{ij}'^{(k)} K_j + c_{ij}'^{(k)} \Omega_j + \sum_{l=j+1}^{N} d_{ij,l}'^{(k)} A_{jl}) + e_i'^{(k)}, \\ &\quad i = 1, 2, \ldots, M, \ k = 1, 2, \ldots, p_i (or\ q_i), \end{aligned} \tag{22}$$

and p_i,q_i are positive integers. Since (21) must be satisfied for any Φ_i, we obtain the following system of equations that must be satisfied:

$$\begin{aligned} \max(h_i^{(1)}, h_i^{(2)}, \ldots, h_i^{(p_i)}) &= \max(h_i'^{(1)}, h_i'^{(2)}, \ldots, h_i'^{(q_i)}), \\ &\quad i = 1, 2, \ldots, M. \end{aligned} \tag{23}$$

Example: In Step 1, we obtained single-max expressions (19) and (20) for the one-soliton solution of the ultradiscrete Lotka-Volterra equation. Using the procedure outlined in Step 2, the left-hand side of the equation (19) leads

to

$$\begin{aligned}\max(&\underline{0}+\max(0,1),\underline{x_1}+\max(0,1,-\Omega_1,-K_1-\Omega_1,-K_1-\Omega_1+1,K_1-\Omega_1+1,\\
&2K_1-2\Omega_1+1,3K_1-2\Omega_1),\underline{2x_1}+\max(-2\Omega_1+1,-\Omega_1,-K_1-2\Omega_1,\\
&-K_1-\Omega_1,-K_1-\Omega_1+1,K_1-3\Omega_1+1,K_1-\Omega_1+1,2K_1-3\Omega_1,\\
&2K_1-2\Omega_1+1,3K_1-3\Omega_1,3K_1-3\Omega_1+1,3K_1-2\Omega_1),\\
&\underline{3x_1}+\max(-2\Omega_1+1,-K_1-2\Omega_1,K_1-3\Omega_1+1,2K_1-4\Omega_1,2K_1-4\Omega_1+1,\\
&2K_1-3\Omega_1,3K_1-3\Omega_1,3K_1-3\Omega_1+1),\underline{4x_1}+\max(2K_1-4\Omega_1,2K_1-4\Omega_1+1)),\end{aligned}$$

and the right-hand side of the equation (20) leads to

$$\begin{aligned}\max(&\underline{0}+\max(0,1),\underline{x_1}+\max(-\Omega_1+1,-2K_1,-K_1+1,K_1-2\Omega_1,K_1-2\Omega_1+1,\\
&K_1-\Omega_1,2K_1-\Omega_1,2K_1-\Omega_1+1),\underline{2x_1}+\max(-2\Omega_1+1,-\Omega_1,\\
&-K_1-2\Omega_1,-K_1-\Omega_1,-K_1-\Omega_1+1,K_1-3\Omega_1+1,K_1-\Omega_1+1,\\
&2K_1-3\Omega_1,2K_1-2\Omega_1+1,3K_1-3\Omega_1,3K_1-3\Omega_1+1,3K_1-2\Omega_1),\\
&\underline{3x_1}+\max(-3\Omega_1,-3\Omega_1+1,K_1-3\Omega_1,K_1-2\Omega_1,K_1-2\Omega_1+1,\\
&2K_1-3\Omega_1+1,3K_1-4\Omega_1+1,4K_1-4\Omega_1),\underline{4x_1}+\max(2K_1-4\Omega_1,2K_1-4\Omega_1+1)).\end{aligned}$$

Consequently, we obtain a system of equations that must be satisfied:

$$\begin{cases}\max(0,1)=\max(0,1),\\
\max(0,1,-\Omega_1,-K_1-\Omega_1,-K_1-\Omega_1+1,K_1-\Omega_1+1,2K_1-2\Omega_1+1,3K_1-2\Omega_1)\\
=\max(-\Omega_1+1,-2K_1,-K_1+1,K_1-2\Omega_1,K_1-2\Omega_1+1,K_1-\Omega_1,2K_1-\Omega_1,\\
\qquad 2K_1-\Omega_1+1),\\
\max(-2\Omega_1+1,-\Omega_1,-K_1-2\Omega_1,-K_1-\Omega_1,-K_1-\Omega_1+1,K_1-3\Omega_1+1,\\
K_1-\Omega_1+1,2K_1-3\Omega_1,2K_1-2\Omega_1+1,3K_1-3\Omega_1,3K_1-3\Omega_1+1,3K_1-2\Omega_1)\\
=\max(-2\Omega_1+1,-\Omega_1,-K_1-2\Omega_1,-K_1-\Omega_1,-K_1-\Omega_1+1,K_1-3\Omega_1+1,\\
K_1-\Omega_1+1,2K_1-3\Omega_1,2K_1-2\Omega_1+1,3K_1-3\Omega_1,3K_1-3\Omega_1+1,3K_1-2\Omega_1),\\
\max(-2\Omega_1+1,-K_1-2\Omega_1,K_1-3\Omega_1+1,2K_1-4\Omega_1,2K_1-4\Omega_1+1,2K_1-3\Omega_1,\\
\qquad 3K_1-3\Omega_1,3K_1-3\Omega_1+1)\\
=\max(-3\Omega_1,-3\Omega_1+1,K_1-3\Omega_1,K_1-2\Omega_1,K_1-2\Omega_1+1,2K_1-3\Omega_1+1,\\
\qquad 3K_1-4\Omega_1+1,4K_1-4\Omega_1),\\
\max(2K_1-4\Omega_1,2K_1-4\Omega_1+1)=\max(2K_1-4\Omega_1,2K_1-4\Omega_1+1).\end{cases}\tag{24}$$

Step 3. *Divide the domain of parameters and evaluate the max equation*

In Step 2, we obtained a system of equations that consisted of parameters K_i, Ω_i and A_{ij} $(i, j = 1, \ldots, N)$. Since Ω_i and $A_{i,j}$ are functions of K_i, we substitute the relations $\Omega_i = \Omega_i(K_i)$ and $A_{ij} = A_{ij}(K_i, K_j)$ into (23). Then, we derive a system of equations that consists of parameters K_i $(i = 1, \ldots, N)$. However, Ω_i and/or A_{ij} often include the max and absolute value functions. Hence, we divide the domain of parameters K_i into regions by solving the max or absolute value equations in Ω_i and/or A_{ij}, and then we evaluate both sides of the max equations in each region.

Example: We consider the one-soliton solution of the ultradiscrete Lotka-Volterra equation. From the dispersion relation of the one-soliton solution (7),

$$\Omega_1 = \max(0, K_1 - 1) - \max(0, -K_1 - 1), \tag{25}$$

we obtain the following three regions for K_1,

$$K_1 < -1, \quad -1 \leq K_1 < 1, \quad K_1 \geq 1. \tag{26}$$

Case 1. For $K_1 < -1$, (25) yields $\Omega_1 = K_1 + 1$. Substituting the value of Ω_1 into (24) and evaluating (24) under the condition $K_1 < -1$,

$$\begin{cases} 1 = 1, \\ -2K_1 = -2K_1, \\ \max(-3K_1 - 2, -2K_1) = \max(-3K_1 - 2, -2K_1), \\ -3K_1 - 2 = -3K_1 - 2, \\ -2K_1 - 3 = -2K_1 - 3. \end{cases} \tag{27}$$

Case 2. For $-1 \leq K_1 < 1$, (25) yields $\Omega_1 = 0$ and (24) reduces to

$$\begin{cases} 1 = 1, \\ \max(1, -K_1 + 1, K_1 + 1, 2K_1 + 1) = \max(1, -K_1 + 1, K_1 + 1, 2K_1 + 1), \\ \max(1, -K_1 + 1, K_1 + 1, 2K_1 + 1, 3K_1 + 1) = \max(1, -K_1 + 1, K_1 + 1, 2K_1 + 1, 3K_1 + 1), \\ \max(1, K_1 + 1, 2K_1 + 1, 3K_1 + 1) = \max(1, K_1 + 1, 2K_1 + 1, 3K_1 + 1), \\ 2K_1 + 1 = 2K_1 + 1. \end{cases} \tag{28}$$

Case 3. For $K_1 \geq 1$, (25) yields $\Omega_1 = K_1 - 1$ and (24) reduces to

$$\begin{cases} 1 = 1, \\ K_1 + 2 = K_1 + 2, \\ \max(4, K_1 + 2) = \max(4, K_1 + 2), \\ 4 = 4, \\ -2K_1 + 5 = -2K_1 + 5. \end{cases} \tag{29}$$

Thus, (24) is satisfied identically for all regions of K_1. Consequently, we verify that (7) is a solution of the equation (6).

The algorithm described above is simple in principle, but it is oftentimes tedious to carry out by hand. In the next section, we describe the *Maple* package `Ultde`, which executes this algorithm.

4 Ultde

The `Ultde` package is currently available for the following one-component max-plus equation:

$$L_1(F_n^m, F_n^{m\pm1}, \ldots, F_{n\pm1}^m, F_{n\pm1}^{m\pm1}, \ldots) = L_2(F_n^m, F_n^{m\pm1}, \ldots, F_{n\pm1}^m, F_{n\pm1}^{m\pm1}, \ldots),$$

where L_1 and L_2 are linear combinations of $F_{n\pm i}^{m\pm j}$ $(i, j = 0, \pm1, \pm2, \ldots)$, which includes the max function. `Ultde` consists of the following procedures.

4.1 The soca operator

The soca operator is one of the main procedures in the `Ultde` package, and it evaluates expressions L_1 and L_2 under the given conditions. It is used with the syntax

soca(L_1: expression, L_2: expression).

If L_1 is equivalent to L_2, the soca generates the message, "The solution is TRUE." This message initiates the procedures below.

4.2 The maxplus operator

The maxplus operator is used to convert the max-plus expression $Expr$ to a single-max form by using the max-plus identities. It is different from the max command in *Maple* and has the following syntax:

maxplus($Expr$: expression).

This procedure corresponds to Step 1 of the above algorithm.

4.3 The geteqxl operator

The geteqxl operator collects the elements of the same phase Xl (that is, a list of coefficients of $x_1, x_2, \ldots, x_N$) in the expression $Expr$ and generates a system of equations that must be satisfied. It uses the following syntax:

geteqxl($Expr$: expression, Xl: list).

This procedure corresponds to Step 2 of the above algorithm.

4.4 The getconds operator

The getconds operator divides the domain of parameters K_1,$K_2, \ldots, K_N$ into regions using the dispersion relation Oml and the phase shift term Al. The syntax is as follows:

getconds(Oml: list, Al: list).

This procedure corresponds to Step 3 of the above algorithm.

4.5 The redueqs operator

The redueqs operator evaluates the expression $Expr$ in each region of parameters K_1,$K_2, \ldots, K_N$ in Cl using the following syntax:

redueqs($Expr$: expression, Cl: list).

This procedure corresponds to Step 3 of the above algorithm. Note that in

our program, we use symmetry among parameters K_i to save the calculation time.

5 Examples

We now use three examples to illustrate our program. The dependent variable F_{n+a}^{m+b}, phase term Ξ_i, wave number K_i, frequency Ω_i and phase shift term A_{ij} used in the above algorithm are expressed as $F(a,b), xi, ki, omi, aij$ in the *Maple* program, respectively.

Example 1. Let us first consider the ultradiscrete Lotka-Volterra equation (6). After loading the package `Ultde` and running the main procedure *soca*, we input the max-plus form of equation (17), and solutions (7), (8) and (9). Outputs show that all solutions identically satisfy equation (17). The running times for one-, two- and three-soliton solutions are 0.5, 20 and 2705 seconds, respectively.

Example 2. Next we consider an ultradiscrete bilinear box-and-ball system(BBS) equation [8],

$$F_{n+1}^{m+1} + F_n^{m-1} = \max(F_{n+1}^m + F_n^m, F_{n+1}^{m-1} + F_n^{m+1} - 1). \tag{30}$$

A multi-soliton solution of (30) is

$$F_n^m = \max_{\mu_i=0,1}\Big(\sum_{1\le i\le N} \mu_i \Xi_i(m,n) - \sum_{1\le i<j\le N} \mu_i\mu_j A_{ij}\Big), \tag{31}$$

where N is a natural number indicating a number of solitons and

$$\begin{aligned} &\Xi_i(m,n) = K_i m - \Omega_i n + \Xi_i^0, \\ &\Omega_i = \frac{1}{2}(|K_i+1| - |K_i-1|), \quad i = 1,2,\dots,N \\ &A_{ij} = |K_i+K_j| - |K_i-K_j|, \quad i\neq j,\ i,j = 1,2,\dots,N. \end{aligned} \tag{32}$$

One-, two- and three-soliton solutions are expressed as follows:
One-soliton solution:

$$\begin{cases} F_n^m = \max(0, \Xi_1), \\ \Xi_1 = K_1 m - \Omega_1 n + \Xi_1^0, \\ \Omega_1 = \frac{1}{2}(|K_1+1| - |K_1-1|). \end{cases} \tag{33}$$

Two-soliton solution:

$$\begin{cases} F_n^m = \max(0, \Xi_1, \Xi_2, \Xi_1 + \Xi_2 - A_{12}), \\ \Xi_i = K_i m - \Omega_i n + \Xi_i^0, \\ \Omega_i = \frac{1}{2}(|K_i + 1| - |K_i - 1|), \quad i = 1, 2 \\ A_{12} = |K_1 + K_2| - |K_1 - K_2|. \end{cases} \tag{34}$$

Three-solution solution:

$$\begin{cases} F_n^m = \max(0, \Xi_1, \Xi_2, \Xi_3, \Xi_1 + \Xi_2 - A_{12}, \Xi_1 + \Xi_3 - A_{13}, \\ \qquad\qquad \Xi_2 + \Xi_3 - A_{23}, \Xi_1 + \Xi_2 + \Xi_3 - A_{12} - A_{13} - A_{23}) \\ \Xi_i = K_i m - \Omega_i n + \Xi_i^0, \\ \Omega_i = \frac{1}{2}(|K_i + 1| - |K_i - 1|), \quad i = 1, 2, 3, \\ A_{ij} = |K_i + K_j| - |K_i - K_j|, \quad i \neq j, \ i, j = 1, 2, 3. \end{cases} \tag{35}$$

After inputting equation (30) and solutions (33), (34) and (35), the outputs show that all solutions identically satisfy equation (30). The running times for one-, two- and three-soliton solutions are 0.2, 2 and 30 seconds, respectively.

Example 3. Finally, we consider the ultradiscrete Burgers equation [9],

$$U_n^{m+1} = U_n^m - \max(-U_{n-1}^m, U_n^m - L) + \max(-U_n^m, U_{n+1}^m - L). \tag{36}$$

By using the transformation from U to F

$$U_n^m = F_{n+1}^m - F_n^m + \frac{L}{2}, \tag{37}$$

we obtain another form of the ultradiscrete Burgers equation,

$$\max(F_{n-1}^m + F_{n+1}^{m+1}, F_{n+1}^m + F_{n+1}^{m+1}) = \max(F_n^m + F_n^{m+1}, F_n^{m+1} + F_{n+2}^m). \tag{38}$$

One-, two- and three-soliton solutions are expressed as follows:
One-soliton solution:

$$\begin{cases} F_n^m = \max(0, \Xi_1), \\ \Xi_1 = K_1 n + \Omega_1 m + \Xi_1^0, \\ \Omega_1 = |K_1|. \end{cases} \tag{39}$$

Two-soliton solution:

$$\begin{cases} F_n^m = \max(0, \Xi_1, \Xi_2), \\ \Xi_i = K_i n + \Omega_i m + \Xi_i^0, \\ \Omega_i = |K_i|, \quad i = 1, 2. \end{cases} \tag{40}$$

Three-soliton solution:

$$\begin{cases} F_n^m = \max(0, \Xi_1, \Xi_2, \Xi_3), \\ \Xi_i = K_i n + \Omega_i m + \Xi_i^0, \\ \Omega_i = |K_i|, \quad i = 1, 2, 3. \end{cases} \tag{41}$$

After inputting the expressions of equation (38) and solutions (39), (40) and (41), the outputs show that all solutions identically satisfy equation (38). The running times for one-, two- and three-soliton solution are 0.1, 0.3, and 0.5 seconds, respectively.

The corresponding *Maple* system responses for these examples are given in the Test run output in Appendix A: note that intermediate output messages are omitted.

6 Conclusions

We have presented an algorithm for verifying ultradiscrete soliton solutions and have developed a corresponding *Maple* program. Using this program, we verified the solutions of the ultradiscrete Lotka-Volterra equation, the ultradiscrete bilinear BBS equation, and the ultradiscrete Burgers equation.

Acknowledgements

This work was supported by the Project of Innovation for Young Researchers in Fujian Province, China, under grant NO. 2005J057.

Appendix A. Test run output

```
# Example 1. Ultradiscrete Lotka-Volterra equation
# Definition of equation
restart:
read"ultde.txt":
Left:=max(F(-1,1)+F(2,2)+F(0,0)+F(1,1)+1,F(0,1)+F(3,2)+F(-1,1)+F(0,0)):
Right:=max(F(-1,0)+F(2,1)+F(0,1)+F(1,2)+1,F(-2,0)+F(1,1)+F(1,2)+F(2,1)):
et:=(i,n,m)-> x||i+n*cat(k,i)-m*cat(om,i):

# 1-soliton solution
```

```
F:=(n,m)-> max(0,et(1,n,m)):
om1:=max(0,k1-1)-max(0,-k1-1):
soca(Left,Right):

     **********  The solution is TRUE ! ***********

# 2-soliton solution
F:=(n,m)-> max(0,et(1,n,m),et(2,n,m),et(1,n,m)+et(2,n,m)+'a12'):
om1:=max(0,k1-1)-max(0,-k1-1):
om2:=max(0,k2-1)-max(0,-k2-1):
a12:=abs(k1-k2)-abs(k1+k2):
soca(Left,Right):

     **********  The solution is TRUE ! ***********

# 3-soliton solution
F:=(n,m)-> max(0,et(1,n,m),et(2,n,m),et(3,n,m),et(1,n,m)+et(2,n,m)+'a12',
               et(1,n,m)+et(3,n,m)+'a13',et(2,n,m)+et(3,n,m)+'a23',
               et(1,n,m)+et(2,n,m)+et(3,n,m)+'a12'+'a13'+'a23'):
om1:=max(0,k1-1)-max(0,-k1-1):
om2:=max(0,k2-1)-max(0,-k2-1):
om3:=max(0,k3-1)-max(0,-k3-1):
a12:=abs(k1-k2)-abs(k1+k2):
a13:=abs(k1-k3)-abs(k1+k3):
a23:=abs(k2-k3)-abs(k2+k3):
soca(Left,Right):

     **********  The solution is TRUE ! ***********


# Examples 2. Ultradiscrete bilinear equation of BBS
# Definition of equation
restart:
read"ultde.txt":
Left:=F(1,1)+F(0,-1):
Right:=max(F(1,0)+F(0,0),F(1,-1)+F(0,1)-1):
et:=(i,n,m)->x||i+m*cat(k,i)-n*cat(om,i):

# 1-soliton solution
F:=(n,m)-> max(0,et(1,n,m)):
om1:=(abs(k1+1)-abs(k1-1))/2:
soca(Left,Right):

     **********  The solution is TRUE ! ***********
```

```
# 2-soliton solution
F:=(n,m)-> max(0,et(1,n,m),et(2,n,m),et(1,n,m)+et(2,n,m)-'a12'):
om1:=(abs(k1+1)-abs(k1-1))/2:
om2:=(abs(k2+1)-abs(k2-1))/2:
a12:=abs(k1+k2)-abs(k1-k2):
soca(Left,Right):

     **********  The solution is TRUE ! ***********

# 3-soliton solution
F:=(n,m)-> max(0,et(1,n,m),et(2,n,m),et(3,n,m),et(1,n,m)+et(2,n,m)-'a12',
               et(1,n,m)+et(3,n,m)-'a13',et(2,n,m)+et(3,n,m)-'a23',
               et(1,n,m)+et(2,n,m)+et(3,n,m)-'a12'-'a13'-'a23'):
om1:=(abs(k1+1)-abs(k1-1))/2:
om2:=(abs(k2+1)-abs(k2-1))/2:
om3:=(abs(k3+1)-abs(k3-1))/2:
a12:=abs(k1+k2)-abs(k1-k2):
a13:=abs(k1+k3)-abs(k1-k3):
a23:=abs(k2+k3)-abs(k2-k3):
soca(Left,Right):

     **********  The solution is TRUE ! ***********


# Examples 3. Ultradiscrete Burgers equation
# Definition of equation
restart:
read"ultde.txt":
Left:=max(F(-1,0)+F(1,1),F(1,0)+F(1,1)):
Right:=max(F(0,0)+F(0,1),F(0,1)+F(2,0)):
et:=(i,n,m)-> x||i+n*cat(k,i)+m*cat(om,i):

# 1-soliton solution
F:=(n,m)-> max(0,et(1,n,m)):
om1:=abs(k1):
soca(Left,Right):

     **********  The solution is TRUE ! ***********

# 2-soliton solution
F:=(n,m)-> max(0,et(1,n,m),et(2,n,m)):
om1:=abs(K1):
om2:=abs(K2):
soca(Left,Right):
```

```
     **********  The solution is TRUE ! ***********

# 3-soliton solution
F:=(n,m)-> max(0,et(1,n,m),et(2,n,m),et(3,n,m)):
om1:=abs(K1):
om2:=abs(K2):
om3:=abs(K2):
soca(Left,Right):

     **********  The solution is TRUE ! ***********
```

References

[1] T. Tokihiro, D. Takahashi, J. Matsukidaira, J. Satsuma, Phys. Rev. Lett. 76 (1996) 3247.

[2] D. Takahashi, J. Satsuma, J. Phys. Soc. Jpn. 59 (1990) 3514.

[3] D. Takahashi, T. Tokihiro, B.Grammaticos, Y. Ohta, A. Ramani, J. Phys. A 30 (1997) 7953.

[4] J. Matsukidaira, J. Satsuma, D. Takahashi, T. Tokihiro, M. Torii, Phys. Lett. A 225 (1997) 287.

[5] T. Tokihiro, D. Takahashi, J. Matsukidaira, J. Phys. A 33 (2000) 607.

[6] R. Hirota, S. Tsujimoto, J. Phys. Soc. Jpn. 64 (1995) 3125.

[7] D. Takahashi, J. Matsukidaira, J. Phys. A 30 (1997) L733.

[8] D. Takahashi, R. Hirota, J. Phys. Soc. Jpn. 76 (2007) 104007.

[9] K. Nishinari, D. Takahashi, J. Phys. A 31 (1998) 5439.