Commun. Theor. Phys. (Beijing, China) **52** (2009) pp. 801–807
© Chinese Physical Society and IOP Publishing Ltd
Vol. 52, No. 5, November 15, 2009

A Maple Package on Symbolic Computation of Hirota Bilinear Form for Nonlinear Equations*

YANG Xu-Dong† and RUAN Hang-Yu

Department of Physics, Ningbo University, Ningbo 315211, China

(Received December 10, 2008; Revised June 2, 2009)

Abstract *An improved algorithm for symbolic computation of Hirota bilinear forms of KdV-type equations with logarithmic transformations is presented. In the algorithm, the general assumption of Hirota bilinear form is successfully reduced based on the property of uniformity in rank. Furthermore, we discard the integral operation in the traditional algorithm. The software package HBFTrans is written in Maple and its running effectiveness is tested by a variety soliton equations.*

PACS numbers: 02.30.Jr, 11.30.-j, 02.70.Wz
Key words: Hirota bilinear form, nonlinear equation, symbolic algebra

1 Introduction

In 1971, Hirota developed a new direct method for constructing the exact multi-soliton solution of the Korteweg-de Vries (KdV) equation.[1] Using this ingenious method, Hirota then obtained the N-soliton solution of the modified Korteweg-de Vries (mKdV),[2] sine-Gordon (SG)[3] and nonlinear Schrödinger (NLS)[4] equations. The key step of the Hirota direct method is to transform the given equation into its Hirota bilinear form (HBF). Once the HBF is obtained, the so-called perturbation expansion method becomes algorithmic. Soon after, Hietarinta designed a program on searching for bilinear equations passing Hirota's three-soliton condition and obtained many new integrable models such as the KdV-type,[5] mKdV-type,[6] SG-type,[7] and NLS-type[8] equations. In 1992, Hereman and Zhuang[9] gave a summary on types of bilinear equations based on the in-depth study of Hietarinta. An extensive study of the Hirota direct method can be found in Ref. [10]. Recently, a reliable algorithm for generating bilinear form of nonlinear partial differential equations (PDE) is described and a *Maple* package Bilinearization is established by Zhou, Fu, and Li.[11] The package can successfully and automatically construct the HBF of many nonlinear PDEs of KdV-type by solving a system of over-determined algebraic equations with respect to the combinatorial coefficients. However, the general ansatz of the bilinear form of nonlinear PDEs in Ref. [11] is so complicated that the running efficiency is relatively lower. In this paper, we make advantage of the scaling property of nonlinear PDEs to simplify the computation, which is partially based on the idea presented by Hereman[12] and Yao[13] in computing the polynomial conservation laws of nonlinear PDEs. Our algorithm can be used to compute the HBF of single nonlinear PDE with a logarithmic transformation. Moreover, the algorithm can be easily implemented in any symbolic language. Here, we also present a software package in *Maple* to automate the tedious computation for the construction of the HBF of nonlinear PDEs.

The paper is organized as follows. In Sec. 2, we outline the algorithm and subroutines in detail. In Sec. 3, several examples are presented. Finally, a short summary is given in the Sec. 4. More details on how to use our package is explained in Appendix.

2 Computation of Hirota Bilinear Form

2.1 *Notation*

(i) *Independent Variables*

$\mathbf{x} = (x_1, x_2, \ldots, x_n)$ denotes n-dimensional spatial and time variables $x_1, x_2, \ldots, x_n$.

(ii) *Dependent Functions*

$\mathbf{u} = (u^1, u^2, \ldots, u^m)$ denotes m-component functions with respect to the variable $\mathbf{x}$.

(iii) *Partial Derivatives*

$$u^i_S = u^i_{s_1,s_2,\ldots,s_n} = u^i_{s_1 x_1 s_2 x_2 \cdots s_n x_n} = \frac{\partial^{s_1+s_2+\cdots+s_n}}{\partial x_1^{s_1} \partial x_2^{s_2} \cdots \partial x_n^{s_n}} u^i$$

denotes all the partial derivatives of the order $k = \sum_{j=1}^{n} s_j$ of u^i with respect to different $s_1, s_2, \ldots, s_n$, and $\mathbf{u}^{(k)}$ denotes a vector whose components are all the partial derivatives of the order 0 up to k of all the u^i. Here, S is a sequence of s_j, and s_j is the partial derivative order of u with respect to x_j.

(iv) *Nonlinear Partial Differential Equations*

$$L^i\big(\mathbf{u}^{(k)}\big) = 0\,, \quad i = 1, 2, \ldots, m\,, \tag{1}$$

*Supported by Scientific Research Fund of Zhejiang Provincial Education Department under Grant No. 20070979, the National Natural Science Foundations of China under Grant Nos. 10675065 and 10735030, the Scientific Research Found of Ningbo University under Grant No. XKL09059, and the K.C.Wong Magana Fund in Ningbo University

†Corresponding author, E-mail: yangxudong@nbu.edu.cn

where L^i is a polynomial of $\mathbf{u}^{(k)}$ denotes a system of m-component PDEs of an order k.

(v) *Functional Transformation*

$$\mathbf{u} = \Phi(\mathbf{f}), \tag{2}$$

where $\mathbf{f} = (f_1, f_2, \ldots, f_l)$ is a group of auxiliary functions depending on variable $\mathbf{x}$, and Φ is a transformation.

(vi) *Hirota Bilinear Form*

$$\sum_{1\le i,j\le l}\Big[\sum_{S}[c_S^{i,j,k}\times D_S]f_i\cdot f_j\Big] = 0\,, \quad k = 1, 2, \ldots, m'\,, \tag{3}$$

denotes the m'-component linear equation, where D_S is the Hirota-D operator defined as

$$\begin{aligned} D_S\, f_i\cdot f_j &= \Big[\prod_{k=1}^{n} D_{x_k}^{s_k}\Big] f_i\cdot f_j \\ &= \Big[\prod_{k=1}^{n}\Big(\frac{\partial}{\partial x_k} - \frac{\partial}{\partial x_k'}\Big)^{s_k}\Big] f_i(\mathbf{x}) f_j(\mathbf{x}')|_{\mathbf{x}=\mathbf{x}'}, \\ & 1\le i,\quad j\le l\,, \end{aligned} \tag{4}$$

and $c_S^{i,\,j,\,k}$ is the corresponding coefficient with respect to the Hirota-D operator D_S.

2.2 *Definition*

For any arbitrary solution $\mathbf{f}$ of Eq. (3), if there exists a functional transformation (2) and its transpositional result $\mathbf{u}$ is also a solution of Eq. (1), then we call Eq. (3) is a HBF of Eq. (1). It is known that the notable KdV equation

$$u_t + 6uu_x + u_{3x} = 0\,, \tag{5}$$

admits three different HBFs. The first one is

$$\Theta\, f\cdot f \equiv (D_x^4 + D_xD_t)\; f\cdot f = 0\,, \tag{6}$$

under a logarithmic transformation

$$u = 2\,\ln(f)_{2x}\,. \tag{7}$$

The other two HBFs are both obtained under a rational transformation, which are in the forms of

$$\begin{aligned} &(D_t + D_x^3)g\cdot f = 0 \quad \text{with} \\ &D_x^2 f\cdot f = 2fg \quad \text{under} \quad u = g/f\,, \end{aligned} \tag{8}$$

and

$$\begin{aligned} &(D_t + D_x^3)g\cdot f = 0 \quad \text{with} \\ &D_x^2 f\cdot f = 2D_x g\cdot f \quad \text{under} \quad u = (g/f)_x\,. \end{aligned} \tag{9}$$

For the complexity of the nonlinear PDEs and the variety of their functional transformations, the process of bilinearization is far from being algorithmic. To show the main idea of our algorithm, for simplicity, we restrict the nonlinear PDEs (1) to

$$L(u^{(k)}) = 0\,, \tag{10}$$

the Hirota bilinear form (3) to

$$\Big[\sum_S c_S\times D_S\Big] f\cdot f = 0\,, \tag{11}$$

and the functional transformation (2) to

$$u = \alpha\times(\ln f)_{sx}\,, \tag{12}$$

where α and c_S are constant coefficients. It is clear that the KdV equation (5) is form invariant under the scaling symmetry transformation

$$(x, t, u)\to(\lambda^{-1}x, \lambda^{-3}t, \lambda^2 u)\,, \tag{13}$$

where λ is a parameter. The scaling invariance, as a special Lie-point symmetry, is an intrinsic property of many integrable nonlinear PDEs. According to (13), without loss of generality, one can introduce the rank of variables as follows:

$$\Omega(u) = 2\,, \quad \Omega(x) = -1\,, \quad \Omega(t) = -3\,. \tag{14}$$

Now, if we define the rank of the partial derivative as

$$\begin{aligned} \Omega(u_S) &= \Omega\Big(\frac{\partial^{s_1+s_2+\cdots+s_n}u}{\partial x_1^{s_1}\partial x_2^{s_2}\cdots\partial x_n^{s_n}}\Big) = \Omega\Big(\frac{u}{\prod_{i=1}^n x_i^{s_i}}\Big) \\ &= \Omega(u) - \sum_{i=1}^{n}[s_i\times\Omega(x_i)]\,, \end{aligned} \tag{15}$$

then two clear and important results should be clarified. The first is that all the monomials in Eq. (5) are *uniform in rank* 5, while the second is that the transformation (7) and the operator Θ in HBF (6) are *uniform in rank* 2 and 4, respectively. It should be emphasized that *uniformity in rank* plays an important role in our algorithm.

2.3 *Algorithm*

The computer algebraic algorithm and routines of constructing the HBF for a single nonlinear PDE under a logarithmic transformation are described in the following steps.

(i) *Determine the weights of variables and parameters*

It is easy to determine the weights of the variables and parameters by using the definition of uniformity in rank. In our algorithm, we construct such a subroutine Weight (*EqList* :: list, *ParaList* :: list) to compute the weights of the variables and parameters, where *EqList* denotes a list of equations and *ParaList* is a list of the parameters with (unknown) weights. Take a (1+1)-dimensional nonlinear PDE

$$w_x + \beta\, w_t - 3w_xw_t - w_{2xt} = 0 \tag{16}$$

as an example, where the parameter β is assumed to have an unknown weight. In this case, one can call the subroutine as Weight($[w_x + \beta\, w_t - 3w_xw_t - w_{2xt} = 0], [\beta]$) to determine the weights of $\{x, t, w, \beta\}$. In detail, the procedure proceeds as follows:

Collecting functions and its dependent variables of *EqList* into the lists *FuncList* and *VarList*, respectively, we have the result

$$\mathit{FuncList} = [w], \quad \mathit{VarList} = [x, t]\,.$$

Constructing the weight list with respect to the *FuncList*, *VarList* and *ParaList*, respectively, we get the result

$$\begin{aligned} &\mathit{WFuncList} = [\Omega_w], \quad \mathit{WVarList} = [\Omega_x, \Omega_t], \\ &\mathit{WParaList} = [\Omega_\beta]\,. \end{aligned}$$

Here, Ω_w, Ω_x, Ω_t, and Ω_β are all elements of an array Ω and denote the weights of w, x, t, and β respectively.

Combining the list *FuncList*, *VarList* and *ParaList*, we derive the set

$$S0 = \{w,\ x,\ t,\ \beta\}\,.$$

Gathering all the monomials of the equation in *EqList*, we derive the set

$$S1 = \{w_x, \beta\, w_t, -3w_x w_t, -w_{2xt}\}\,.$$

Selecting all the partial derivatives of the equation, we get the set

$$S2 = \{w_x, w_t, w_{2xt}\}\,.$$

Replacing each element in $S2$ by the formula $\partial^i w/\partial x^i \to \partial^i w/\partial x^i = w/x^i$, we obtain the set

$$S3 = \{w_x = w/x, w_t = w/t, w_{2xt} = w/x^2/t\}\,.$$

Substituting $S3$ into $S1$, we derive a new set

$$S4 = \{w/x, \beta w/t, -3w^2/x/t, -w/x^2/t\}\,.$$

Rewriting each element in $S4$ through the formula $h = \ln(h)$, we obtain

$$\begin{aligned} S5 = \{&\ln(w) - \ln(x), \ln(\beta) + \ln(w) - \ln(t),\\ &2\ln(w) - \ln(x) - \ln(t) + \ln(3) + \pi\, I,\\ &\ln(w) - 2\ln(x) - \ln(t) + \pi\, I\}. \end{aligned}$$

Removing all the monomials without weight in $S5$, we have the result

$$\begin{aligned} S6 = \{&\ln(w) - \ln(x), \ln(\beta) + \ln(w) - \ln(t),\\ &2\ln(w) - \ln(x) - \ln(t),\\ &\ln(w) - 2\ln(x) - \ln(t)\}. \end{aligned}$$

Replacing each element in $S0$ by letting $h \to h = \exp(\Omega_h)$, then we have the set

$$\begin{aligned} S7 = \{&w = \exp(\Omega_w), x = \exp(\Omega_x),\\ &t = \exp(\Omega_t), \beta = \exp(\Omega_\beta)\}. \end{aligned}$$

Substituting $S7$ into $S6$, we derive a new set

$$\begin{aligned} S8 = \{&\Omega_w - \Omega_x, \Omega_\beta + \Omega_w - \Omega_t,\\ &2\,\Omega_w - \Omega_x - \Omega_t, \Omega_w - 2\,\Omega_x - \Omega_t\}. \end{aligned}$$

Constructing the rank equation with the condition of *uniform in rank* and adding the general assumption $\Omega_x = -1$ yield

$$\begin{aligned} S9 = \{&\Omega_w - \Omega_x = \Omega_\beta + \Omega_w - \Omega_t,\\ &\Omega_w - \Omega_x = 2\,\Omega_w - \Omega_x - \Omega_t,\\ &\Omega_w - \Omega_x = \Omega_w - 2\,\Omega_x - \Omega_t,\ \Omega_x = -1\}. \end{aligned}$$

Solving the equations in $S9$ generates the solution

$$\mathrm{WSol} = \{\Omega_w = 1, \Omega_x = -1, \Omega_t = 1, \Omega_\beta = 2\}\,.$$

According to WSol, four cases should be considered:

(a) WSol is NULL, namely, the equation is not uniform in rank. The program prompts the information and then ends the process.

(b) WSol with a free weight. It requires the user to enter a value and then goes on.

(c) WSol with a fractal weight. In this case, one should multiply each weight in WSol with the lease common multiple of the denominators in the fractal weights and then goes on.

(d) otherwise, goes on.

Substituting the solution WSol into S8, we obtain the weight of the equation

$$WEq = 2\,.$$

Substituting the solution WSol into *WFuncList*, *WVarList*, and *WParaList*, respectively, we have the result

$$\begin{aligned} &WFuncList = [1], \quad WVarList = [-1, 1],\\ &WParaList = [2]. \end{aligned}$$

It is noted that our package is incapable in the case that a function or parameter have a negative weights. That is to say, the values in the list *WFuncList* and *WParaList* should be positive, otherwise the program shows the information and then ends the process.

(ii) *Determine the general form of functional transformation*

Using the property of uniformity in rank, the functional transformation (12) becomes

$$u = \alpha \times (\ln f)_{sx}\,,\ \ s = -\Omega(u)/\Omega(x)\,. \tag{17}$$

Now, we use the weights of the variables and construct a procedure called GenFuncTrans(*FuncList* :: *list*, *VarList* :: *list*, *WFuncList*::list, *WVarList*::list) to determine the general form of the functional transformation. Its algorithm is described as

Begin
iFunc:= op(1,*FuncList*); ♯ Fetch out w into iFunc;
iVar:= op(1, *VarList*); ♯ Fetch out x into iVar;
iWFunc:= op(1, *WFuncList*); ♯ Fetch out the weight of w into iWFunc;
iWVar:= op(1, *WVarList*); ♯ Fetch out the weight of x into iWVar;
iNum:= -iWFunc/iWVar; ♯ Compute $s = -\Omega_w\,/\,\Omega_x$;
If type(iNum,posint) **Then**
FuncTrans:=iFunc= α ∗ diff(f(op(*VarList*)), iVar$iNum));
♯ f is the name of an auxiliary function;
Return(*FuncTrans*);
Else
Error('Fail in computing the general form of functional transformation!');
Fi;
End

Remarks:

Maple function op(*n*,*pList*): return the *n*th element of a list *pList*.

Maple function type(*n*,*posint*): check *n* is a positive integer number or not.

Maple function diff(*pExpr*,*x*$*n*): compute the *n*th-order derivative of *pExpr* w.r.t. *x*.

Calling the subroutine GenFuncTrans([[w],[x,t],[1],[-1,1]), one can obtain the functional transformation of Eq. (16)

$$w = \alpha \times (\ln f)_x = \alpha \times f_x/f\,. \tag{18}$$

(iii) *Construct a general form of HBF*

Substituting the obtained functional transformation (18) into Eq. (16), one can yield a complicated mulit-linear equation with respect to the auxiliary function f

$$\begin{aligned}&(3\alpha-6)f_x^3f_t-(3\alpha-6)ff_x(f_xf_{xt}+f_{2x}f_t)-f^3(f_{2x}+\beta f_{xt}-f_{3xt})\\&+f^2[(3\alpha-3)f_{2x}f_{xt}+f_x^2-f_{3x}f_t-3f_xf_{2xt}+\beta f_xf_t]\,=\,0\,.\end{aligned} \tag{19}$$

According to the algorithm in Ref. [11], one should convert the multi-linear equation (19) into a quadratic equation before constructing the general form of HBF. Applying the HBM method[14] or WTC method,[15] one can get

$$\alpha = 2\,. \tag{20}$$

Then substituting $\alpha = 2$ into Eq. (19) yields

$$\begin{aligned}&f(f_{2x}+\beta f_{xt}-f_{3xt})-3f_{2x}f_{xt}+f_x^2\\&-f_{3x}f_t-3f_xf_{2xt}+\beta f_xf_t\,=\,0\,.\end{aligned} \tag{21}$$

If Eq. (21) is not in a quadratic form, then it is required to integrate the equation with respect of x once or several times. Hereafter, using the general ansatz (3.8) in Ref. [11], one can obtain the general form of HBF

$$\left[a_{31}D_x^3D_t+a_{11}D_xD_t+a_{20}D_x^2\right]f\cdot f=0\,, \tag{22}$$

where a_{31}, a_{11}, and a_{20} are constants to be determined later. However, to convert a multi-linear equation to a quadratic equation is usually rather tedious. Here, we introduce a new procedure to construct the general form of HBF avoiding such a tedious action. First, we rewrite Eq. (11) in the form of

$$\begin{aligned}\Big[\sum_S c_SD_S\Big]f\cdot f&=\Big[\sum_{S,Q}\Big(c_{S,Q}\prod_{k=1}^r p_k^{q_k}\prod_{k=1}^n D_{x_k}^{s_k}\Big)\Big]f\cdot f\\&=0\,,\end{aligned} \tag{23}$$

where p_k denotes a parameter with a weight, q_k is the power of p_k, Q is a sequence of $q_1, q_2, \ldots, q_r$, and $c_{S,Q}$ is a constant coefficient without any weight. Then, we utilize the property of uniformity in rank and the obtained weights of the variables and parameters to construct the general form of HBF.

For a given rank *iRank*, the procedure GetCombination(*VarList*::list, *ParaList*::list, *WVarList*::list, *WParaList*::list, *iRank*::posint, *MLEq*::equation) will automatically insert all the possible combinations into a set in the form of QS = $\{[[q_1, q_2, \ldots, q_r], [s_1, s_2, \ldots s_n]]\}$ which satisfies

$$\sum_{k=1}^r[q_k\times\Omega(p_k)]-\sum_{k=1}^n[s_k\times\Omega(x_k)]=iRank\,, \tag{24}$$

and

$$s_k\le \text{differder}(MLEq, x_k)\,, \tag{25}$$

where *MLEq* is a multi-linear PDE, and differder is a *maple* function which evaluates the differential order of an algebraic expression.

Taking *MLEq* (19), *VarList*=$[x,t]$, *ParaList*=$[\beta]$, *WVarList*=$[-1,1]$, *WParaList*=[2], and *iRank*=2 as an example, the procedure proceeds in detail as follows:

Computing the differential order of the equation *MLEq* (19) with respect to *VarList*, we have

$$\text{MaxDiffOrderList} = [3,1]\,.$$

Constructing all the possible combinations [s_1, s_2, ..., s_k, ..., s_n] based on MaxDiffOrderList and the condition (25) produces a list

$$\begin{aligned}\text{iVarList} = [&[0,0],[0,1],[1,0],[1,1],\\&[2,0],[2,1],[3,0],[3,1]].\end{aligned}$$

Computing the sum of weights for each combination in iVarList by using the formula $\sum_{k=1}^n[s_k\times\Omega(x_k)]$ with *WVarList*, we get

$$\text{iWVarList} = [0,1,-1,0,-2,-1,-3,-2]\,.$$

Constructing all the possible combinations [q_1, q_2, ..., q_k, ..., q_r] under the constraint $q_k\times\Omega(p_k)\le iRank$ yields a list

$$\text{iParaList} = [[0],[1]]\,.$$

Computing the sum of weights for each combination in iParaList by the formula $\sum_{k=1}^r[q_k\times\Omega(p_k)]$ with *WParaList*, we obtain

$$\text{iWParaList} = [0,2]\,.$$

For each element A_i in iWVarList, if A_i is a positive even number, and there exists an element B_j in iWParaList satisfying $B_j-A_i = iRank$, then union the i-th element of iVarList into a set. The result is

$$rVarSet = \{[1,1],[2,0],[3,1]\}\,.$$

According to the set *rVarSet*, one can easily construct an HBF which is similar to Eq. (22). In general, under the constraints of (24) and (25), the sum of the Hirota-D operator of the general form of HBF is much less than that in Ref. [11], which makes our algorithm more efficient. Taking the KdV equation as an example, the general form of HBF under Eqs. (24)–(25), and $iRank = 4$ reads

$$[c_1D_x^4+c_2D_xD_t]f\cdot f\,=\,0\,,$$

while the general ansatz of HBF in Ref. [11] becomes

$$[a_{40}D_x^4+a_{31}D_x^3D_t+a_{20}D_x^2+a_{11}D_xD_t]f\cdot f\,=\,0\,.$$

The difference between two packages are illustrated through some PDEs as shown in Table 1.

Table 1 The running efficiency of the package *HBFTrans*.

Nonlinear partial differential equation	Terms of general ansatz of HBF (*Bilinearization*/*HBFTrans*)	Total running time (*HBFTrans*)
KdV	4/2	0.282
SK	6/2	0.313
Boussinesq	6/5	0.891
SWW	3/3	0.266
BSK	9/3	0.453
KP	7/3	0.344
(3+1)-D KdV	8/5	0.484

After the function GetCombination(*VarList*::list, *ParaList*::list, *WVarList*::list, *WParaList*::list, *iRank*::posint, *MLEq*::equation) is programmed, we can proceed. Using the obtained *rVarSet* with the definition (4), we introduce another subroutine GenHBF(*VarList*::list, *rVarSet*::set) to compute the general form of HBF both in Hirota-D operator form and in its expanded quadratic form. The algorithm is described in the following way.

Begin
iFunc1:=f(op(*VarList*)); iFunc2:=f(op(*VarList*));
♯ f is the name of an auxiliary function;
For ii **From** 1 to nops(*VarList*) **Do** ♯ variable substitution
iFunc2:=subs(op(ii, *VarList*)=Omega[ii], iFunc2);
Od;
tExpr:=iFunc1 * iFunc2;
rQExpr:=0; *rDExpr*:=0;
For ii **From** 1 to nops(*rVarSet*) **Do**
nVarList:=op(ii,*rVarSet*); iQExpr:=tExpr; iDExpr:=c[ii];
For jj **From** 1 to nops(*VarList*) **Do** ♯ compute derivative
ivar:=op(jj, *VarList*); ♯ derivative variable;
idnum:=op(jj, nVarList); ♯ derivative order;
iDExpr:=iDExpr*D[ivar]^idnum;
For kk **From** 1 to idnum **Do**
iQExpr:=diff(iQExpr,ivar)-diff(iQExpr,Omega[jj]);
Od;
Od;
rQExpr:=*rQExpr*+c[ii]*iQExpr; *rDExpr*:=*rDExpr*+iDExpr;
Od;
For ii **From** 1 to nops(*VarList*) **Do** ♯ variable inverse substitution
rQExpr:=subs(Omega[ii]=op(ii, *VarList*), *rQExpr*);
Od;
Return(*rQExpr*,*rDExpr*);
End

Calling the function GenHBF([x,t],{[1,1],[2,0],[3,1]}), one can easily obtain

$$\mathrm{rDExpr} = c_1 D_x D_t + c_2 D_x^2 + c_3 D_x^3 D_t\,, \tag{26}$$

and

$$\begin{aligned}\mathrm{rQExpr} = {} & 2c_1 f f_{3xt} - 6c_1 f_x f_{2xt} + 6c_1 f_{2x} f_{xt} \\ & - 2c_1 f_{3x} f_t + 2c_2 f f_{xt} - 2c_2 f_x f_t \\ & + 2c_3 f f_{2x} - 2c_3 f_x^2\,. \end{aligned} \tag{27}$$

(iv) *Determine the coefficients*

In the previous subsection, we give out a subroutine to construct the general form of HBF for a given rank. In this subsection, by using the definition of the HBF, we construct a subroutine GetHBF(*MLEq*::equation,*rQExpr*) to determine the coefficients c_S in Eq. (11) and α in Eq. (12). Concretely, the procedure proceeds as follows:

Fetching out the maximum differential term of *rQExpr* (27), we have

$$\mathrm{MaxDiffTerm} = f_{3xt}\,.$$

Solving equation rQExpr = 0 with respect to MaxDiffTerm, we obtain

$$\begin{aligned}\mathrm{iSol} = {} & (3c_1 f_x f_{2xt} - 3c_1 f_{2x} f_{xt} + c_1 f_{3x} f_t \\ & - c_2 f f_{xt} + c_2 f_x f_t - c_3 f f_{2x} + c_3 f_x^2)/c_1/f.\end{aligned}$$

Substituting MaxDiffTerm = iSol into the multi-linear equation *MLEq* (19) and then collecting all the coefficients of the product of potential function and its derivatives, yields an equation

$$\mathrm{iEq} = \{a_3\beta + a_1, -a_3\beta - a_1, -6a_3 + 3a_3\alpha,$$

$$-a_2-a_3, a_3+a_2, 6a_3-3a_3\alpha\}.$$

Solving the equations in iEq leads to a nontrivial solution

$$\text{rSol}=\{\alpha=2, \beta=\beta, a_1=-a_3\beta, a_2=-a_3, a_3=a_3\}.$$

By substituting the solution rSol into Eqs. (26) and (18), one can obtain the final result $(\beta D_xD_t+D_x^2-D_x^3D_t)f\cdot f=0$ under the transformation $w=2(\ln f)_x$.

In the traditional algorithm, one should integrate iExpr once or several times to get its quadratic form. However, the integral operation is a weakness of most symbolic computation systems. Furthermore, we find it is not necessary and also waste of time. In our algorithm, by directly substituting the general form of HBF into *MLEq*, we successfully discard the integral operation, which greatly improve the running efficiency.

Up to now, the key steps of our algorithm have been discussed in the above subsections. To make more clear the relationship among these key steps, we design a main function HBFTMain(*EqList*::list, *ParaList*::list, *iRank*::posint). The parameter *iRank* is optional and its default value is a maximal even number less than *WEq*. In detail, the procedure is as follows:

Using subroutine Weight with parameter *EqList* and *ParaList*, we obtain the return value *FuncList*, *VarList*, *WFuncList*, *WVarList*, *WParaList*, and *WEq*.

If the optional parameter *iRank* is not given, then we assign a maximal even number less than *WEq* to *iRank*.

Calling function GenFuncTrans with parameter *FuncList*, *VarList*, *WFuncList*, and *WVarList*, we have the functional transformation *FuncTrans*.

Substituting the obtained *FuncTrans* into *EqList*, we obtain a mulit-linear equation *MLEq*.

Using the procedure GetCombination with parameter *VarList*, *ParaList*, *WVarList*, *WParaList*, *iRank*, and *MLEq*, we obtain a combination *rVarSet*.

Calling function GenTrans with parameter *VarList* and *rVarSet*, we have the general form of HBF both in Hirota-D operator form *rDExpr* and in its expanded quadratic form *rQExpr*.

Using the subroutine GetHBF with parameter *MLEq* and *rQExpr*, we have a solution of the combinatorial coefficients.

Substituting the obtained results into *rDExpr* and *FuncTrans*, the HBF of the studied equation is finally obtained.

3 Applications

Example 1 The first example we consider is the Sawada–Kotera (SK) equation

$$u_t+45u^2u_x-15u_xu_{2x}-15uu_{3x}+u_{5x}=0, \tag{28}$$

which is a (1+1)-dimensional nonlinear PDE of an order 5. First, HBFTrans generates the weights of the variables

$$\Omega(u)=2, \quad \Omega(x)=-1, \quad \Omega(t)=-5.$$

Then HBFTrans reports that the SK equation possesses an HBF as $[D_tD_x+D_x^6]f\cdot f=0$ under the functional transformation $u=-(\ln f)_{2x}$.

Example 2 Here, we consider a nonlinear PDE without uniformity in rank, the Boussinesq equation

$$u_{2t}-u_{2x}-3(u^2)_{2x}-u_{4x}=0. \tag{29}$$

First, HBFTrans denotes that the system (29) is not uniform in rank. If we add a parameter with a weight to the system and call the program in the form of HBFTMain($[u_{2t}-\beta u_{2x}-3(u^2)_{2x}-u_{4x}=0], [\beta]$), then the outputs are

$$\Omega(u)=2, \quad \Omega(x)=-1, \quad \Omega(t)=-2, \quad \Omega(\beta)=2,$$

and $[D_t^2-\beta D_x^2-D_x^4]f\cdot f=0$ with $u=2(\ln f)_{2x}$. By setting $\beta=1$, the final HBF of the Boussinesq equation is obtained.

Example 3 Let us consider another nonlinear PDE with an integration term, the shallow water wave (SWW) equation

$$u_t+u_x-3uu_t-u_{2xt}+3u_x\int_x^\infty u_t dx=0. \tag{30}$$

First, by substituting $u=w_x$ and using the boundary condition $w_t|_{x\to\infty}=0$, we convert Eq. (30) into a differential form

$$w_{xt}+w_{2x}-3w_xw_{xt}-w_{3xt}-3w_{2x}w_t=0. \tag{31}$$

Obviously, Eq. (31) is not uniform in rank. Therefore, one should call the program in the form of HBFTMain($[\beta w_{xt}+w_{2x}-3w_xw_{xt}-w_{3xt}-3w_{2x}w_t=0], [\beta]$). Then the program derives the weight of variables

$$\Omega(u)=1, \quad \Omega(x)=-1, \quad \Omega(t)=1, \quad \Omega(\beta)=2,$$

and the HBF $(D_tD_x^3-\beta D_tD_x-D_x^2)f\cdot f=0$ with a transformation $w=2(\ln f)_x$.

Example 4 To show the advantage of our package HBFTrans, let us solve a more complicated nonlinear PDE with two integration terms, the Bidirectional SK (BSK) equation

$$\begin{aligned}&-5\int_x^\infty u_{2t}dx+5u_{2xt}-15uu_t+15u_x\int_x^\infty u_t dx\\&-45u^2u_x+15u_xu_{2x}+15uu_{3x}-u_{5x}=0.\end{aligned} \tag{32}$$

Similarly, we first convert Eq. (32) into a differential form by substituting $u=w_x$ and using the boundary conditions $w_t|_{x\to\infty}=0$ and $w_{2t}|_{x\to\infty}=0$, which reads

$$\begin{aligned}\text{BSKN}\equiv{}&5w_{2t}+5w_{3xt}-15(w_xw_t+w_x^3-w_xw_{3x})_x-w_{6x}\\&=0.\end{aligned} \tag{33}$$

Now we call the program HBFTMain([BSKN], [ ]), and then obtain $\Omega(w)=1$, $\Omega(x)=-1$, $\Omega(t)=-3$, and $[5D_t^2+5D_tD_x^3-D_x^6]f\cdot f=0$ with a transformation $w=-2(\ln f)_x$.

Example 5 The previous four example are all (1+1)-dimensional PDEs. Actually, our program is also effective in solving high dimensional nonlinear equations. In this example, we work on a (2+1)-dimensional nonlinear PDE, the KP equation

$$(u_t+6uu_x+u_{xxx})_x-3u_{yy}=0. \tag{34}$$

If we call the program as HBFTMain([KP], [ ]), then HBFTrans produces the weights of variables are

$$\Omega(u) = 2, \quad \Omega(x) = -1, \quad \Omega(y) = -2, \quad \Omega(t) = -3\,,$$

and then the KP equation has no HBF with the default rank 6. However, if we call the program as HBFTMain([KP], [ ], 4), then we have the right result $(D_tD_x + D_x^4 - 3D_y^2)f \cdot f = 0$ with $u = 2(\ln f)_{xx}$.

Example 6 Consider the (3+1)-dimensional KdV equation

$$u_t + 6u_xu_y + u_{2xy} + u_{4xz} + 60u_x^2u_z + 10u_{3x}u_z + 20u_xu_{2xz} = 0\,. \tag{35}$$

The outputs of HBFTrans are

$$\Omega(u) = 1, \quad \Omega(x) = -1, \quad \Omega(y) = \Omega(t) + 2,$$
$$\Omega(z) = \Omega(t) + 4, \quad \Omega(t) = \Omega(t),$$

which mean that Eq. (35) has a free weight. Then we just take the weights of the variables to be

$$\Omega(u) = 1, \quad \Omega(x) = -1, \quad \Omega(y) = -3,$$
$$\Omega(z) = -1, \quad \Omega(t) = -5.$$

In this case, we have the result $(D_x^3D_y + D_x^5D_z + D_xD_t)f \cdot f = 0$ with $u = (\ln f)_x$. The running effectiveness of our package HBFTrans is listed in Table 1.

4 Summary

In this paper, we present a new algorithm for symbolic computation of HBF of a single nonlinear equation with a logarithmic transformation and the package in *Maple*. Using our package, several classical nonlinear equations are successfully tested and the results are listed. It is clear that our algorithm based on uniformity in rank can be extended in parallel to the KdV equation (5) under a rational transformation for both the cases of (8) and (9) by fixing $\Omega(f) = 0$ and determining $\Omega(g)$ through the uniformity of rank of the functional transformation. However, it is difficult to present a uniform algorithm since it might involve so many types of functional transformations such as rational, logarithmic, bi-logarithmic, and their combinations, or even some unknown complicated transformations. Anyway, a more generalized algorithm of computing HBF of nonlinear PDEs is still worthy of further study.

Acknowledgements

We thank the referees for their constructive suggestions.

Appendix

The usage of HBFTrans

The package HBFTrans can work in *Maple* 9, 10, and even more higher versions. Its main procedure is denoted as HBFTMain. We take Eq. (16) as an example to show how to use HBFTrans.

To start, one proceeds as follows:

[>with(HBFTrans);

$$[HBFTMain]$$

[>alias(u=u(x,t),f=f(x,t));

$$u, f$$

[>HBFTMain([diff(u,x)+beta*diff(u,t)-diff(u,x,x,t)-3* diff(u,x)* diff(u,t)=0],[beta]);

The input equation is

$$\frac{\partial}{\partial x}u + \beta\frac{\partial}{\partial t}u - \frac{\partial^3}{\partial x^2\partial t} - 3\,\frac{\partial}{\partial x}u\,\frac{\partial}{\partial t}u \;=\; 0\,.$$

The weights of the variables are:

$$\{\Omega_\beta = 2, \Omega_t = 1, \Omega_x = -1, \Omega_u = 1\}\,.$$

Seeking the Hirota bilinear form on rank $2\cdots$.

The ansatz of functional transformation is

$$u = \frac{\alpha\frac{\partial}{\partial x}f}{f}\,.$$

The ansatz of Hirota bilinear form is

$$(c_1D_xD_t + c_2D_x^2 + c_3D_x^3D_t)@[f..f] = 0\,.$$

The solution set for the coefficients is

$$\{c_2 = -c_3, c_1 = -\beta c_3, \alpha = 2, \beta = \beta, c_3 = c_3\}\,.$$

The given equation possesses an HBF

$$(\beta D_xD_t + D_x^2 - D_x^3D_t)@[f..f] = 0\,,$$

under a transformation

$$u = \frac{2\frac{\partial}{\partial x}f}{f}\,.$$

References

[1] R. Hirota, Phys. Rev. Lett. **27** (1971) 1192.
[2] R. Hirota, J. Phys. Soc. Jpn. **33** (1972) 1456.
[3] R. Hirota, J. Phys. Soc. Jpn. **33** (1972) 1459.
[4] R. Hirota, J. Math. Phys. **14** (1973) 805.
[5] J. Hietarinta, J. Math. Phys. **28** (1987) 1732.
[6] J. Hietarinta, J. Math. Phys. **28** (1987) 2094.
[7] J. Hietarinta, J. Math. Phys. **28** (1987) 2586.
[8] J. Hietarinta, J. Math. Phys. **29** (1988) 628.
[9] W. Hereman and W. Zhuang, *Computational and Applied Mathematics II: Differential Equations*, eds.: W.F. Ames and P.J. van der Houwen, Elsevier, Amsterdam (1992).
[10] R. Hirota, *The Direct Method in Soliton Theory*, Cambridge University Press, Cambridge (2004).
[11] Z.J. Zhou, J.Z. Fu, and Z.B. Li, Appl. Math. Comput. **183** (2006) 872.
[12] W. Hereman, Int. J. Quantum. Chem. **106** (2006) 278.
[13] R.X. Yao and Z.B. Li, Appl. Math. Comput. **173** (2006) 616.
[14] M.L. Wang, Y.B. Zhou, and Z.B. Li, Phys. Lett. A **216** (1996) 67.
[15] J. Weiss, M. Tabor, and G. Carnevale, J. Math. Phys. **24** (1983) 522.