



CIRM, Luminy

Sept. 28–Oct.2, 2009

Some Results on Integrable Algorithms

XING-BIAO HU

*ICMSEC, AMSS, Chinese Academy of Sciences
P.O.Box 2719, China, 100080*

hxb@lsec.cc.ac.cn

This is joint work with Yi HE, Hon-Wah TAM and Satoshi Tsujimoto

Outline



A brief introduction



A general form of sequence transformations and triangular recursion schemes

Title: Some Results on Integrable Algorithms

Keywords: Integrable algorithms

- ”可積分アルゴリズム”, in Y. Nakamura’s book: 2006 Functionality of Integrable Systems, Kyoritsu Shuppan Co., Tokyo, Japan (in Japanese).
- Y. Nakamura, ”Why so accurate is an integrable SVC algorithm ?”, 数理解析研究所講究録1473 巻2006 年41-61

Questions:

- Q1: What is an integrable algorithm?
- Q2: Why interesting?
- Q3: How to find integrable algorithms?

Q1: an algorithm \longrightarrow a partial difference equation \longrightarrow integrable equation \longrightarrow an integrable algorithm

Q1 →

- What is integrability for an equation?

Definition of Integrability:

- For a finite Hamiltonian system ($2m$ variables), we say it is completely integrable if it admits m constants of the motion $F_i, i = 1, \dots, m$, which are independent and in involution under Poisson bracket associated with the Hamiltonian structure and level surface defined by the intersection of surfaces $F_i = c_i$ is compact and connected.
- Infinite-dimensional case: **No** unified definition

working definitions

- Lax-integrable: We say an equation is integrable if it can be represented as a compatibility condition of a pair of linear equations or a commutation relation of a pair of linear operators.

KdV: $u_t + 6uu_x + u_{xxx} = 0$

$$\psi_{xx} + u\psi = \lambda\psi$$

$$\psi_t = u_x\psi - (2u + 4\lambda)\psi_x$$

$$\psi_{xxt} = \psi_{txx} \implies u_t + 6uu_x + u_{xxx} = 0$$

- Symmetries and conservation laws
- Bi-Hamiltonian structures
- Painlevé property
- Bäcklund transformations
- N-soliton solutions
- C-integrable
-

Examples of integrable algorithms:

- ε algorithm:

$$(\varepsilon_{k+1}^{(n)} - \varepsilon_{k-1}^{(n+1)})(\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}) = 1$$

$$\varepsilon_{-1}^{(n)} = 0, \quad \varepsilon_0^{(n)} = S_n \quad (n = 0, 1, 2, \dots)$$

Discrete potential KdV

- ρ algorithm:

$$(\rho_{k+1}^{(n)} - \rho_{k-1}^{(n+1)})(\rho_k^{(n+1)} - \rho_k^{(n)}) = k$$

$$\rho_0^{(n)} = 0, \quad \rho_1^{(n)} = S_n \quad (n = 0, 1, 2, \dots)$$

Continuous limit: cylindrical KdV

- η algorithm:

$$\eta_{k+1}^{(n)} - \eta_{k-1}^{(n+1)} = \frac{1}{\eta_k^{(n+1)}} - \frac{1}{\eta_k^{(n)}}$$

$$\eta_0^{(n)} = 0, \quad \eta_1^{(n)} = \Delta S_{n-1} \quad (n = 0, 1, 2, \dots), \quad S_{-1} = 0$$

Q2: Why are we interested in integrable algorithms?

- They are attractive! Some famous algorithms are integrable ϵ -algorithm, η -algorithm, ρ algorithm, qd algorithm
- They have several significant applications
For example, qd algorithm: solving matrix eigenvalue problems, algebraic equations, the BCH-Goppa decoding problem and a sorting problem...
- They have nice properties and structures and numerical performance

$$(\epsilon_{k+1}^{(n)} - \epsilon_{k-1}^{(n+1)})(\epsilon_k^{(n+1)} - \epsilon_k^{(n)}) = 1$$

$$\epsilon_{-1}^{(n)} = 0, \quad \epsilon_0^{(n)} = S_n \quad (n = 0, 1, 2, \dots)$$

Hankel determinant solution:

$$\epsilon_{2k}^{(n)} = \frac{\begin{vmatrix} S_n & S_{n+1} & \cdots & S_{n+k} \\ S_{n+1} & S_{n+2} & \cdots & S_{n+k+1} \\ \vdots & \vdots & & \vdots \\ S_{n+k} & S_{n+k+1} & \cdots & S_{n+2k} \end{vmatrix}}{\begin{vmatrix} \Delta^2 S_n & \Delta^2 S_{n+1} & \cdots & \Delta^2 S_{n+k-1} \\ \Delta^2 S_{n+1} & \Delta^2 S_{n+2} & \cdots & \Delta^2 S_{n+k} \\ \vdots & \vdots & & \vdots \\ \Delta^2 S_{n+k-1} & \Delta^2 S_{n+k} & \cdots & \Delta^2 S_{n+2k-2} \end{vmatrix}}$$

$$\varepsilon_{2k+1}^{(n)} = \frac{\begin{vmatrix} \Delta^3 S_n & \Delta^3 S_{n+1} & \cdots & \Delta^3 S_{n+k-1} \\ \Delta^3 S_{n+1} & \Delta^3 S_{n+2} & \cdots & \Delta^3 S_{n+k} \\ \vdots & \vdots & & \vdots \\ \Delta^3 S_{n+k-1} & \Delta^3 S_{n+k} & \cdots & \Delta^3 S_{n+2k-2} \end{vmatrix}}{\begin{vmatrix} \Delta S_n & \Delta S_{n+1} & \cdots & \Delta S_{n+k} \\ \Delta S_{n+1} & \Delta S_{n+2} & \cdots & \Delta S_{n+k+1} \\ \vdots & \vdots & & \vdots \\ \Delta S_{n+k} & \Delta S_{n+k+1} & \cdots & \Delta S_{n+2k} \end{vmatrix}}$$

Comments and Remarks on importance of this subject

1. R. Hirota et al, Mathematics and Computers in Simulation 37(1994)371-383

”The study of difference scheme of integrable systems is currently the focus of intense activities.”

2. C. Brezinski, Convergence acceleration during the 20th century, Journal of Computational and Applied Mathematics 122 (2000) 1-21

”...In particular, the connection between convergence acceleration algorithms and continuous and discrete integrable systems brings a different and fresh look to both domains and could be of benefit to them...”

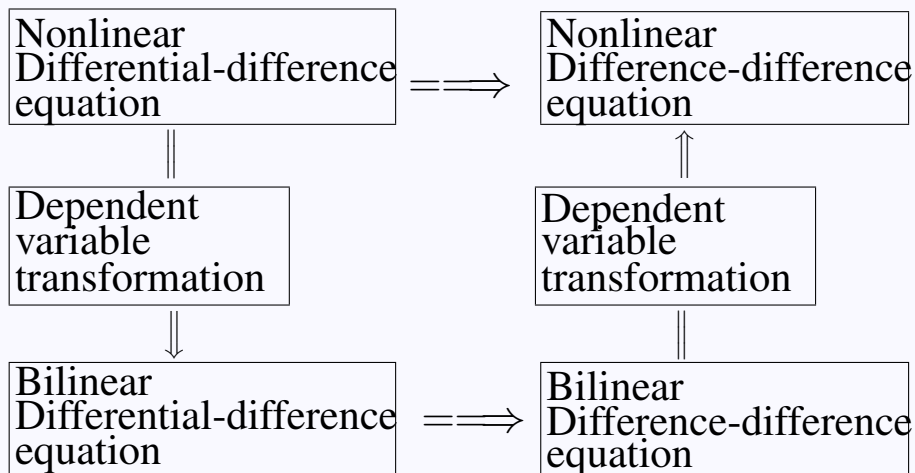
3. Moody T. Chu., Linear algebra algorithms as dynamical systems, Acta Numerica (2008), pp. 1 – 86

”...it is truly remarkable that diverse topics, such as soliton theory, integrable systems, continuous fraction, τ -functions, orthogonal polynomials, the Sylvester identity, moments, and Hankel determinants, can all play together, interwine, and eventually lead to the fact abstractly, but literally, that the eigenvalues and singular values of a given matrix can be expressed as the limit of some closed-form formulas!”

- Q3: How to find integrable algorithms?

Integrable discretizations of integrable systems

Hirota's discretization



time-discretization of the Lotka-Volterra lattice

- Step 1

Lotka-Volterra lattice

$$u_{n,t} = u_n(u_{n-1} - u_{n+1}) \quad (1)$$

$$u(n) = \frac{f(n+2)f(n-1)}{f(n)f(n+1)}, \quad \text{Bilinear form}$$

$$[D_t e^{\frac{1}{2}D_n} + e^{\frac{3}{2}D_n} - e^{\frac{1}{2}D_n}] f_n \cdot f_n = 0 \quad (2)$$

$$D_t a \cdot b = a_t b - a b_t, \quad e^{D_n} a_n \cdot b_n = a_{n+1} b_{n-1}.$$

- Step 2

$$f_n^{m+1} f_{n+1}^m - f_n^m f_{n+1}^{m+1} - \delta [f_{n-1}^m f_{n+2}^{m+1} - f_n^m f_{n+1}^{m+1}] = 0 \quad (3)$$

where $t = m\delta$. when $\delta \rightarrow 0$, (3) is reduced to Lotka-Volterra lattice (2).

- Step 3

$$\text{Dependent variable transformation } u_n^m = \frac{f_{n-1}^m f_{n+2}^{m+1}}{f_{m,n} f_{n+1}^{m+1}},$$

Nonlinear form

$$u_n^{m+1} - u_n^m = \hat{\delta} (u_n^m u_{n-1}^m - u_n^{m+1} u_{n+1}^{m+1}) \quad (4)$$

$$\text{where } \hat{\delta} = \frac{\delta}{1-\delta}.$$

Stage 1: Integrable algorithms \longrightarrow Continuous integrable systems

Stage 2: Fully discrete integrable systems \longleftrightarrow Integrable algorithms

Stage 3: Integrable algorithms \longrightarrow new continuous integrable systems

Stage 4: Integrable discretizations \longrightarrow Integrable numerical algorithms

Example 1 The discrete Lotka-Volterra system with variable step-size: singular value computation (M. Iwasaki and Y. Nakamura *Inverse Problems* 20(2004)553-563)

Example 2 The discrete relativistic Toda molecule equation: a Padé approximation algorithm (Y. Minesaki and Y. Nakamura, *Numerical Algorithms* 27(2001)219-235)

Example 3 The discrete mKdV equation: mKdV algorithm to solve a class of algebraic equations (A. Mukaihira and Y. Nakamura *Inverse Problems* 16(2000)413-424)

A general form of sequence transformations and triangular recursion schemes

Known results:

(C. Brezinski, G. Walz, J. Comput. Appl. Math. 34 (1991) 361-383.)

- sequence transformations of the form

$$T_k^{(n)} = \sum_{i=0}^k \alpha_{k,i}^{(n)} S_{n+i} . \quad (5)$$

- a triangular recursion scheme

$$T_k^{(n)} = a_k^{(n)} T_{k-1}^{(n)} + b_k^{(n)} T_{k-1}^{(n+1)} \quad (6)$$

- E-transformation

- E-algorithm: a) Brezinski-Havie E-algorithm b) Ford-Sidi E-algorithm

Our goals:

- 1) to generalize C. Brezinski and G. Walz's results
- 2) to generalize Brezinski-Havie E-algorithm, Ford-Sidi E-algorithm and so on

- a general form of sequence transformations of the form

$$T_k^{(n)} = \sum_{i=0}^k \alpha_{k,i}^{(n)} S_{n+kp+iJ}, \quad (7)$$

where $J > 0, p > 0$ are two integers.

- the recursion scheme

$$T_k^{(n)} = a_k^{(n)} T_{k-1}^{(n+p)} + b_k^{(n)} T_{k-1}^{(n+q)} \quad (8)$$

where q is an integer and $J = q - p$.

C. Brezinski, M. Redivo Zaglia 《Extrapolation Methods: Theory and Practice》 1991

Following C. Brezinski, G. Walz's idea, we can obtain the following results:

Theorem 1. Let $\{T_k^{(n)}\}$ be the sequence transformations obtained by the recursion scheme (8). There exist real or complex numbers $\{\alpha_{k,j}^{(n)}\}$ such that

$$T_k^{(n)} = \sum_{i=0}^k \alpha_{k,i}^{(n)} S_{n+kp+iJ},$$

with

$$\alpha_{0,0}^{(n)} = 1, \tag{9}$$

$$\alpha_{k,0}^{(n)} = a_k^{(n)} \alpha_{k-1,0}^{(n+p)}, \tag{10}$$

$$\alpha_{k,i}^{(n)} = a_k^{(n)} \alpha_{k-1,i}^{(n+p)} + b_k^{(n)} \alpha_{k-1,i-1}^{(n+q)}, \quad i = 1, 2, \dots, k-1, \tag{11}$$

$$\alpha_{k,k}^{(n)} = b_k^{(n)} \alpha_{k-1,k-1}^{(n+q)}. \tag{12}$$

In addition, for all n and k , $\sum_{i=0}^k \alpha_{k,i}^{(n)}$ is independent of n , say $\sum_{i=0}^k \alpha_{k,i}^{(n)} = \alpha_k$, if and only if for all n and k , $a_k^{(n)} + b_k^{(n)}$ is independent of n , say $a_k^{(n)} + b_k^{(n)} = \gamma_k$, and $\alpha_0 = \gamma_0 = 1$ and $\alpha_k = \prod_{i=0}^k \gamma_i$.

Theorem 2 Let $T_k^{(n)}(\sigma)$ be a reference functional of the form $T_k^{(n)}(\sigma) = \sum_{i=0}^k \alpha_{k,i}^{(n)} \sigma(z_{n+kp+iJ})$. Furthermore, assume that $\sigma_0, \sigma_1, \dots, \sigma_k$ be elements of $\mathcal{A}(E, F)$ such that

$$\begin{vmatrix} \sigma_0(z_{n+kp}) & \sigma_0(z_{n+kp+J}) & \cdots & \sigma_0(z_{n+kp+kJ}) \\ \sigma_1(z_{n+kp}) & \sigma_1(z_{n+kp+J}) & \cdots & \sigma_1(z_{n+kp+kJ}) \\ \vdots & \vdots & & \vdots \\ \sigma_k(z_{n+kp}) & \sigma_k(z_{n+kp+J}) & \cdots & \sigma_k(z_{n+kp+kJ}) \end{vmatrix} \neq 0,$$

and $T_k^{(n)}(\sigma)$ satisfies

$$T_k^{(n)}(\sigma_0) = w_k^{(n)}, \quad (13)$$

$$T_k^{(n)}(\sigma_i) = 0, \quad i = 1, 2, \dots, k, \quad (14)$$

where $\{w_k^{(n)}\}$ are arbitrary nonzero real or complex numbers.

Then the transformations $\{T_k^{(n)}(\sigma)\}$ have the presentation

$$T_k^{(n)}(\sigma) = \frac{\begin{vmatrix} \sigma(z_{n+kp}) & \sigma(z_{n+kp+J}) & \cdots & \sigma(z_{n+kp+kJ}) \\ \sigma_1(z_{n+kp}) & \sigma_1(z_{n+kp+J}) & \cdots & \sigma_1(z_{n+kp+kJ}) \\ \vdots & \vdots & & \vdots \\ \sigma_k(z_{n+kp}) & \sigma_k(z_{n+kp+J}) & \cdots & \sigma_k(z_{n+kp+kJ}) \end{vmatrix}}{\begin{vmatrix} \sigma_0(z_{n+kp}) & \sigma_0(z_{n+kp+J}) & \cdots & \sigma_0(z_{n+kp+kJ}) \\ \sigma_1(z_{n+kp}) & \sigma_1(z_{n+kp+J}) & \cdots & \sigma_1(z_{n+kp+kJ}) \\ \vdots & \vdots & & \vdots \\ \sigma_k(z_{n+kp}) & \sigma_k(z_{n+kp+J}) & \cdots & \sigma_k(z_{n+kp+kJ}) \end{vmatrix}} \cdot w_k^{(n)} \quad (15)$$

Theorem 3 Let $\{T_k^{(n)}(\sigma)\}$ be the reference functional as

$$T_k^{(n)}(\sigma) = \sum_{i=0}^k \alpha_{k,i}^{(n)} \sigma(z_{n+kp+iJ})$$

and satisfy the conditions (13) and (14) in Theorem 3.1. If $\forall k, n$ and for $i = 0, 1$, assume

$$\begin{vmatrix} \sigma_i(z_n) & \sigma_i(z_{n+J}) & \cdots & \sigma_i(z_{n+kJ}) \\ \sigma_{i+1}(z_n) & \sigma_{i+1}(z_{n+J}) & \cdots & \sigma_{i+1}(z_{n+kJ}) \\ \vdots & \vdots & & \vdots \\ \sigma_{i+k}(z_n) & \sigma_{i+k}(z_{n+J}) & \cdots & \sigma_{i+k}(z_{n+kJ}) \end{vmatrix} \neq 0, \quad (16)$$

then $\{T_k^{(n)}(\sigma)\}$ can be computed recursively by

$$T_k^{(n)}(\sigma) = a_k^{(n)} T_{k-1}^{(n+p)}(\sigma) + b_k^{(n)} T_{k-1}^{(n+q)}(\sigma) \quad (17)$$

with $T_0^{(n)}(\sigma) = \sigma(z_n)$ and

$$a_k^{(n)} = w_k^{(n)} T_{k-1}^{(n+q)}(\sigma_k) / d_k^{(n)} \quad (18)$$

$$b_k^{(n)} = -w_k^{(n)} T_{k-1}^{(n+p)}(\sigma_k) / d_k^{(n)} \quad (19)$$

$$d_k^{(n)} = w_{k-1}^{(n+p)} T_{k-1}^{(n+q)}(\sigma_k) - w_{k-1}^{(n+q)} T_{k-1}^{(n+p)}(\sigma_k) \quad (20)$$

Theorem 4 A necessary and sufficient condition that $\forall n, T_k^{(n)}(\sigma)/w_k^{(n)} = S$ is that

$$\forall n, \sigma(z_n) = S\sigma_0(z_n) + c_1\sigma_1(z_n) + \cdots + c_k\sigma_k(z_n).$$

Theorem 5 If $\forall m \geq 1$

$$\begin{vmatrix} T_k^{(n+mp)}(\sigma_0) & T_k^{(n+mp+J)}(\sigma_0) & \cdots & T_k^{(n+mp+mJ)}(\sigma_0) \\ T_k^{(n+mp)}(\sigma_{k+1}) & T_k^{(n+mp+J)}(\sigma_{k+1}) & \cdots & T_k^{(n+mp+mJ)}(\sigma_{k+1}) \\ \vdots & \vdots & & \vdots \\ T_k^{(n+mp)}(\sigma_{k+m}) & T_k^{(n+mp+J)}(\sigma_{k+m}) & \cdots & T_k^{(n+mp+mJ)}(\sigma_{k+m}) \end{vmatrix} \neq 0$$

then $T_{k+m}^{(n)}(\sigma)$ can be computed by

$$T_{k+m}^{(n)}(\sigma) = w_{k+m}^{(n)} \frac{\begin{vmatrix} T_k^{(n+mp)}(\sigma) & T_k^{(n+mp+J)}(\sigma) & \cdots & T_k^{(n+mp+mJ)}(\sigma) \\ T_k^{(n+mp)}(\sigma_{k+1}) & T_k^{(n+mp+J)}(\sigma_{k+1}) & \cdots & T_k^{(n+mp+mJ)}(\sigma_{k+1}) \\ \vdots & \vdots & & \vdots \\ T_k^{(n+mp)}(\sigma_{k+m}) & T_k^{(n+mp+J)}(\sigma_{k+m}) & \cdots & T_k^{(n+mp+mJ)}(\sigma_{k+m}) \end{vmatrix}}{\begin{vmatrix} T_k^{(n+mp)}(\sigma_0) & T_k^{(n+mp+J)}(\sigma_0) & \cdots & T_k^{(n+mp+mJ)}(\sigma_0) \\ T_k^{(n+mp)}(\sigma_{k+1}) & T_k^{(n+mp+J)}(\sigma_{k+1}) & \cdots & T_k^{(n+mp+mJ)}(\sigma_{k+1}) \\ \vdots & \vdots & & \vdots \\ T_k^{(n+mp)}(\sigma_{k+m}) & T_k^{(n+mp+J)}(\sigma_{k+m}) & \cdots & T_k^{(n+mp+mJ)}(\sigma_{k+m}) \end{vmatrix}}$$

$$T_k^{(n)}(\sigma) = \frac{\begin{vmatrix} \sigma(z_{n+kp}) & \sigma(z_{n+kp+J}) & \cdots & \sigma(z_{n+kp+kJ}) \\ \sigma_1(z_{n+kp}) & \sigma_1(z_{n+kp+J}) & \cdots & \sigma_1(z_{n+kp+kJ}) \\ \vdots & \vdots & & \vdots \\ \sigma_k(z_{n+kp}) & \sigma_k(z_{n+kp+J}) & \cdots & \sigma_k(z_{n+kp+kJ}) \end{vmatrix}}{\begin{vmatrix} \sigma_0(z_{n+kp}) & \sigma_0(z_{n+kp+J}) & \cdots & \sigma_0(z_{n+kp+kJ}) \\ \sigma_1(z_{n+kp}) & \sigma_1(z_{n+kp+J}) & \cdots & \sigma_1(z_{n+kp+kJ}) \\ \vdots & \vdots & & \vdots \\ \sigma_k(z_{n+kp}) & \sigma_k(z_{n+kp+J}) & \cdots & \sigma_k(z_{n+kp+kJ}) \end{vmatrix}} \cdot w_k^{(n)} \quad (21)$$

Choose $\sigma(z_n) = S_n$, $\sigma_0(z_n) = 1$, $\sigma_i(z_n) = g_i(n)$, $i = 1, 2, \dots, k$, $w_k^{(n)} = 1$. We get a general E-transformation as

$$\bar{E}_k^{(n)} = \frac{\begin{vmatrix} S_{n+kp} & S_{n+kp+J} & \cdots & S_{n+kp+kJ} \\ g_1(n+kp) & g_1(n+kp+J) & \cdots & g_1(n+kp+kJ) \\ \vdots & \vdots & & \vdots \\ g_k(n+kp) & g_k(n+kp+J) & \cdots & g_k(n+kp+kJ) \end{vmatrix}}{\begin{vmatrix} \Delta_J g_1(n+kp) & \Delta_J g_1(n+kp+J) & \cdots & \Delta_J g_1(n+kp+(k-1)J) \\ \Delta_J g_2(n+kp) & \Delta_J g_2(n+kp+J) & \cdots & \Delta_J g_2(n+kp+(k-1)J) \\ \vdots & \vdots & & \vdots \\ \Delta_J g_k(n+kp) & \Delta_J g_k(n+kp+J) & \cdots & \Delta_J g_k(n+kp+(k-1)J) \end{vmatrix}}$$

where the operator Δ_m is defined by $\Delta_m S_n = S_{n+m} - S_n$. and $J = q - p$

Following Brezinski's idea, a general Brezinski-Havie E-algorithm is proposed to implement the general E-transformation

$$\begin{aligned}\bar{E}_0^{(n)} &= S_n, & n = 0, 1, \dots, \\ g_{0,i}^{(n)} &= g_i(n), & n = 0, 1, \dots, \quad i = 1, 2, \dots, \\ \bar{E}_k^{(n)} &= \bar{E}_{k-1}^{(n+p)} - \frac{\bar{E}_{k-1}^{(n+q)} - \bar{E}_{k-1}^{(n+p)}}{g_{k-1,k}^{(n+q)} - g_{k-1,k}^{(n+p)}} g_{k-1,k}^{(n+p)}, \\ g_{k,i}^{(n)} &= g_{k-1,i}^{(n+p)} - \frac{g_{k-1,i}^{(n+q)} - g_{k-1,i}^{(n+p)}}{g_{k-1,k}^{(n+q)} - g_{k-1,k}^{(n+p)}} g_{k-1,k}^{(n+p)}, & i = k + 1, k + 2, \dots,\end{aligned}$$

where the auxiliary quantities $\{g_{k,i}^{(n)}\}$ are defined by

$$g_{k,i}^{(n)} = \frac{\begin{vmatrix} g_i(n + kp) & g_i(n + kp + J) & \cdots & g_i(n + kp + kJ) \\ g_1(n + kp) & g_1(n + kp + J) & \cdots & g_1(n + kp + kJ) \\ \vdots & \vdots & & \vdots \\ g_k(n + kp) & g_k(n + kp + J) & \cdots & g_k(n + kp + kJ) \end{vmatrix}}{\begin{vmatrix} \Delta_J g_1(n + kp) & \Delta_J g_1(n + kp + J) & \cdots & \Delta_J g_1(n + kp + (k - 1)J) \\ \Delta_J g_2(n + kp) & \Delta_J g_2(n + kp + J) & \cdots & \Delta_J g_2(n + kp + (k - 1)J) \\ \vdots & \vdots & & \vdots \\ \Delta_J g_k(n + kp) & \Delta_J g_k(n + kp + J) & \cdots & \Delta_J g_k(n + kp + (k - 1)J) \end{vmatrix}}$$

Following (W. F. Ford, A. Sidi, SIAM J. Numer. Anal., 24 (1987) 1212-1232.), a general Ford-Sidi E-algorithm can be obtained:

For sequence $u = (u_0, u_1, \dots)$, set

$$\Psi_k^{(n)}(u) = \frac{\begin{vmatrix} u_{n+kp} & u_{n+kp+J} & \cdots & u_{n+kp+kJ} \\ g_1(n+kp) & g_1(n+kp+J) & \cdots & g_1(n+kp+kJ) \\ \vdots & \vdots & & \vdots \\ g_k(n+kp) & g_k(n+kp+J) & \cdots & g_k(n+kp+kJ) \end{vmatrix}}{\begin{vmatrix} g_{k+1}(n+kp) & g_{k+1}(n+kp+J) & \cdots & g_{k+1}(n+kp+kJ) \\ g_1(n+kp) & g_1(n+kp+J) & \cdots & g_1(n+kp+kJ) \\ \vdots & \vdots & & \vdots \\ g_k(n+kp) & g_k(n+kp+J) & \cdots & g_k(n+kp+kJ) \end{vmatrix}}$$

then we have

$$\bar{E}_k^{(n)} = \frac{\Psi_k^{(n)}(S)}{\Psi_k^{(n)}(1)}$$

and $\{\Psi_k^{(n)}(u)\}$ can be computed by

$$\Psi_k^{(n)}(u) = \frac{\Psi_{k-1}^{(n+p)}(u) - \Psi_{k-1}^{(n+q)}(u)}{\Psi_{k-1}^{(n+p)}(g_{k+1}) - \Psi_{k-1}^{(n+q)}(g_{k+1})}$$

We propose a general hungry type E-algorithm to compute the general E-transformation

$$\frac{V_{k+1}^{n+q} - V_{k+1}^{n+p}}{V_k^{n+q} - V_k^{n+p}} = \frac{V_{k+M+1}^n V_{k-M}^{n+p+q}}{V_k^{n+p} V_k^{n+q}}, \quad (22)$$

with initial values

$$\begin{aligned} V_0^n &= V_1^n = \dots = V_{M-1}^n = 1, \\ V_M^n &= g_1(n), V_{M+i}^n = \frac{g_{i+1}(n)}{g_i(n)}, \quad i = 1, 2, \dots, M-2 \\ V_{2M-1}^n &= \frac{S_n}{g_{M-1}(n)}, V_{2M}^n = \frac{\Delta_J g_1(n+p)}{S_n} \end{aligned}$$

and we have

$$\bar{E}_{M-1}^{(n)} = (-1)^{M-1} V_{(M-1)(M+1)+M}^n.$$

By variable transformation

$$V_k^n = \frac{\tau_{k+1}^n}{\tau_k^n}$$

we get the bilinear form of the equation (22)

$$\tau_{k+M+1}^n \tau_{k-M}^{n+p+q} = \tau_k^{n+p} \tau_{k+1}^{n+q} - \tau_k^{n+q} \tau_{k+1}^{n+p} \quad (23)$$

When $p = 0, q = 1$ algorithm (22) reduces to hungry type E-algorithm and equation (23) reduces to the bilinear form of discrete hungry Lotka-Volterra equation.

(S. Tsujimoto, in *Soliton Theory and Its Applications* J. Satsuma, ed.), University of Tokyo, Japan, (1995) pp. 53-56 (In Japanese)

Theorem 6 Define the function f_m^n as

$$f_m^n = f_{k(M+1)+l}^n = A_{k,l}^n(\varphi; M), \quad 0 \leq l \leq M$$

where the determinants $A_{k,l}^n(\varphi; M)$ are given by

$$A_{k,l}^n(\varphi; M) = \begin{cases} 0 & (k < 0), \\ 1 & (k = 0), \\ \det|h_{i,j}^k(j+l-1; M)|_{1 \leq i,j \leq k} & (k > 0) \end{cases}$$

and the elements $h_{i,j}^k$ are defined as

$$h_{i,j}^k(j+l-1; M) = \begin{cases} \varphi_{n_j+1}(n+(k-1)p+(i-1)J) & \text{if } t_j = 0 \\ \Delta_J \varphi_{n_j+1}(n+kp+(i+t_j-2)J) & \text{else} \end{cases} \\ (j+l-1 = t_j M + n_j, \quad 0 \leq n_j < M)$$

Then f_k^n satisfy the bilinear equation

$$f_{k+M+1}^n f_{k-M}^{n+p+q} = f_k^{n+p} f_{k+1}^{n+q} - f_k^{n+q} f_{k+1}^{n+p} \quad (24)$$

with initial values

$$f_0^n = f_1^n = \dots = f_M^n = 1, \\ f_{M+i}^n = \varphi_i(n), \quad i = 1, 2, \dots, M$$

The first particular case of general E-transformation: general Richardson extrapolation process

choosing $g_i(n) = x_n^i, i = 1, 2, \dots, k$, we derive a general transformation corresponding to the Richardson extrapolation process

$$\bar{T}_k^{(n)} = \frac{\begin{vmatrix} S_{n+kp} & S_{n+kp+J} & \cdots & S_{n+kp+kJ} \\ x_{n+kp} & x_{n+kp+J} & \cdots & x_{n+kp+kJ} \\ \vdots & \vdots & & \vdots \\ x_{n+kp}^k & x_{n+kp+J}^k & \cdots & x_{n+kp+kJ}^k \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_{n+kp} & x_{n+kp+J} & \cdots & x_{n+kp+kJ} \\ \vdots & \vdots & & \vdots \\ x_{n+kp}^k & x_{n+kp+J}^k & \cdots & x_{n+kp+kJ}^k \end{vmatrix}},$$

It is obvious that $g_{k-1,k}^{(n)} = (-1)^{k-1} x_{n+kp} \cdot x_{n+kp+J} \cdot \dots \cdot x_{n+kp+(k-1)J}$.

We obtain the general Richardson extrapolation process

$$\begin{aligned} \bar{T}_0^{(n)} &= S_n \\ \bar{T}_k^{(n)} &= \frac{x_{n+kp+kJ} \bar{T}_{k-1}^{(n+p)} - x_{n+kp} \bar{T}_{k-1}^{(n+h)}}{x_{n+kp+kJ} - x_{n+kp}} \end{aligned}$$

$\bar{T}_k^{(n)}$ is the value at the point 0 of the interpolation polynomial $P_k^{(n)}$, of degree at most k , which satisfies

$$P_k^{(n)}(x_{n+kp+iJ}) = S_{n+kp+iJ}, \quad i = 0, 1, \dots, k$$

The second particular case of general E-transformation: general G-transformation

From the general E-transformation, by choosing $g_i(n) = x_{n+(i-1)R}$, $i = 1, 2, \dots, k$, we derive a general G-transformation

$$\bar{G}_k^{(n)} = \frac{\begin{vmatrix} S_{n+kp} & S_{n+kp+J} & \cdots & S_{n+kp+kJ} \\ x_{n+kp} & x_{n+kp+J} & \cdots & x_{n+kp+kJ} \\ \vdots & \vdots & & \vdots \\ x_{n+kp+(k-1)R} & x_{n+kp+(k-1)R+J} & \cdots & x_{n+kp+(k-1)R+kJ} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_{n+kp} & x_{n+kp+J} & \cdots & x_{n+kp+kJ} \\ \vdots & \vdots & & \vdots \\ x_{n+kp+(k-1)R} & x_{n+kp+(k-1)R+J} & \cdots & x_{n+kp+(k-1)R+kJ} \end{vmatrix}},$$

where R is a positive integer.

We study the special case when $R = J$.

Set

$$r_k^{(n)} = \frac{\begin{vmatrix} x_{n+kp} & x_{n+kp+J} & \cdots & S_{n+kp+(k-1)J} \\ x_{n+kp+J} & x_{n+kp+2J} & \cdots & x_{n+kp+kJ} \\ \vdots & \vdots & & \vdots \\ x_{n+kp+(k-1)J} & x_{n+kp+kJ} & \cdots & x_{n+kp+(2k-2)J} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_{n+kp} & x_{n+kp+J} & \cdots & S_{n+kp+(k-1)J} \\ \vdots & \vdots & & \vdots \\ x_{n+kp+(k-2)J} & x_{n+kp+(k-1)J} & \cdots & x_{n+kp+(2k-3)J} \end{vmatrix}},$$

$$s_k^{(n)} = \frac{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_{n+kp} & x_{n+kp+J} & \cdots & x_{n+kp+kJ} \\ \vdots & \vdots & & \vdots \\ x_{n+kp+(k-1)J} & x_{n+kp+kJ} & \cdots & x_{n+kp+(2k-1)J} \end{vmatrix}}{\begin{vmatrix} x_{n+kp} & x_{n+kp+J} & \cdots & S_{n+kp+(k-1)J} \\ x_{n+kp+J} & x_{n+kp+2J} & \cdots & x_{n+kp+kJ} \\ \vdots & \vdots & & \vdots \\ x_{n+kp+(k-1)J} & x_{n+kp+kJ} & \cdots & x_{n+kp+(2k-2)J} \end{vmatrix}}.$$

Obviously, we have $r_k^{(n)} = (-1)^{k-1} g_{k-1,k}^{(n)}$.

By applying determinantal identities, the $\{r_k^{(n)}\}$ and $\{s_k^{(n)}\}$ can be recursively computed by the general rs-algorithm

$$s_0^{(n)} = 1, \quad r_1^{(n)} = x_{n+p}$$

$$s_{k+1}^{(n)} = s_k^{(n+q)} \left(\frac{r_{k+1}^{(n+J)}}{r_{k+1}^{(n)}} - 1 \right)$$

$$r_{k+1}^{(n)} = r_k^{(n+q)} \left(\frac{s_k^{(n+q)}}{s_k^{(n+p)}} - 1 \right)$$

$$\bar{G}_0^{(n)} = S_n$$

$$\bar{G}_k^{(n)} = \bar{G}_{k-1}^{(n+p)} - \frac{\bar{G}_{k-1}^{(n+q)} - \bar{G}_{k-1}^{(n+p)}}{r_k^{(n+q)} - r_k^{(n+p)}} r_k^{(n+p)}$$

Define the Hankel determinants

$$H_k(S_n) \equiv \begin{vmatrix} S_{n+kp} & S_{n+kp+J} & \cdots & S_{n+kp+(k-1)J} \\ S_{n+kp+J} & S_{n+kp+2J} & \cdots & S_{n+kp+kJ} \\ \vdots & \vdots & & \vdots \\ S_{n+kp+(k-1)J} & S_{n+kp+kJ} & \cdots & S_{n+kp+2(k-1)J} \end{vmatrix},$$

$$H_0(S_n) \equiv 1.$$

Then this algorithm is related to a general qd-algorithm:

$$e_0^{(n)} = 0, \quad q_1^{(n)} = \frac{x_{n+q}}{x_{n+p}}$$

$$e_k^{(n)} = q_k^{(n+q)} + e_{k-1}^{(n+q)} - q_k^{(n+p)}$$

$$q_{k+1}^{(n)} = \frac{e_k^{(n+J)} q_k^{(n+q)}}{e_k^{(n)}}$$

The third particular case of general E-transformation: general Shanks' transformation

$$g_i(n) = \Delta_R S_{n+(i-1)R}$$

where R is a positive integer.

—→ a general Shanks' transformation:

$$e_k^{(n)} = \frac{\begin{vmatrix} S_{n+kp} & S_{n+kp+J} & \cdots & S_{n+kp+kJ} \\ \Delta_R S_{n+kp} & \Delta_R S_{n+kp+J} & \cdots & \Delta_R S_{n+kp+kJ} \\ \vdots & \vdots & & \vdots \\ \Delta_R S_{n+kp+(k-1)R} & \Delta_R S_{n+kp+(k-1)R+J} & \cdots & \Delta_R S_{n+kp+(k-1)R+kT} \end{vmatrix}}{\begin{vmatrix} \Delta_J \Delta_R S_{n+kp} & \Delta_J \Delta_R S_{n+kp+J} & \cdots & \Delta_J \Delta_R S_{n+kp+(k-1)J} \\ \Delta_J \Delta_R S_{n+kp+R} & \Delta_J \Delta_R S_{n+kp+R+J} & \cdots & \Delta_J \Delta_R S_{n+kp+R+(k-1)J} \\ \vdots & \vdots & & \vdots \\ \Delta_J \Delta_R S_{n+kp+(k-1)R} & \Delta_J \Delta_R S_{n+kp+(k-1)R+J} & \cdots & \Delta_J \Delta_R S_{n+kp+(k-1)R+(k-1)J} \end{vmatrix}}$$

where $R > 0$ is a integer and $J = q - p$.

When $R = J$, we propose another recursion algorithm to implement this particular Shanks' transformation as follows

$$\bar{\epsilon}_{-1}^{(n)} = 0, \quad \bar{\epsilon}_0^{(n)} = S_n, \quad (25)$$

$$\bar{\epsilon}_{2k+1}^{(n)} = \bar{\epsilon}_{2k-1}^{(n+q)} + \frac{1}{\bar{\epsilon}_{2k}^{(n+J)} - \bar{\epsilon}_{2k}^{(n)}}, \quad (26)$$

$$\bar{\epsilon}_{2k+2}^{(n)} = \bar{\epsilon}_{2k}^{(n+q)} + \frac{1}{\bar{\epsilon}_{2k+1}^{(n+q)} - \bar{\epsilon}_{2k+1}^{(n+p)}}, \quad (27)$$

and we have $e_k^{(n)} = \bar{\epsilon}_{2k}^{(n)}$.

From equations (26) and (27), we get the confluent ϵ -algorithm

$$\epsilon_{k+1}(t) = \epsilon_{k-1}(t) + \frac{1}{\epsilon'_k(t)}.$$

P. Wynn, Arch. Math 11:223-36, 1960

Numerical example. We consider the sequence

$$S_n = 2^{n+1} \sin\left(\frac{\pi}{2^{n+1}}\right),$$

which converges to $S = \pi$.

By application of algorithm (26) and (27), we get the following results

Case 1. Let $q = 1$, $p = 0$.

n	$\epsilon_0^{(n)}$	$\epsilon_2^{(n)}$	$\epsilon_4^{(n)}$	$\epsilon_6^{(n)}$
0	2.000000	3.152682	3.14159026817	3.141592653609323
1	2.828427	3.142231	3.14159261749	3.141592653589869
2	3.061467	3.141632	3.14159265303	3.141592653589791
3	3.121445	3.141595	3.14159265358	

Case 2. Let $q = 3$, $p = 0$.

n	$\epsilon_2^{(n)}$	$\epsilon_4^{(n)}$	$\epsilon_6^{(n)}$
0	3.141634284	3.141592653589654	3.141592653589793
1	3.141595127	3.141592653589791	
2	3.141592807		
3	3.141592663		

Case 3. Let $q = 3$, $p = 1$.

n	$\epsilon_2^{(n)}$	$\epsilon_4^{(n)}$	$\epsilon_6^{(n)}$
0	3.141632286	3.141592653589656	3.141592653589793
1	3.141595097	3.141592653589792	
2	3.141592806		
3	3.141592663		

Conclusion

- Sequence transformation $T_k^{(n)} = \sum_{i=0}^k \alpha_{k,i}^{(n)} S_{n+kp+iJ}$
- Recursion scheme $T_k^{(n)} = a_k^{(n)} T_{k-1}^{(n+p)} + b_k^{(n)} T_{k-1}^{(n+q)}$.
- General E-transformation:
 - (a) General Brezinski-Havie E-algorithm
 - (b) General Ford-Sidi E-algorithm
 - (c) General hungry-type E-algorithm
- Three special cases of general E-transformation:
 - 1) general Richardson extrapolation process
 - 2) General G-transformation
—→ An extended rs algorithm ↔ an extended qd algorithm
 - 3) a general Shanks' transformation
—→ An extended ϵ algorithm

Further work in progress:

- Other special cases of general E-transformation
- Vector case

Thank you!



Email:hxb@lsec.cc.ac.cn