Performance evaluation of an efficient double-double BLAS1 function with error-free transformation and its application to explicit extrapolation methods

Tomonori Kouya
Shizuoka Institute of Science and Technology
kouya.tomonori@sist.ac.jp

ARITH26
June 10 – 12, 2019@Kyoto, Japan

Outline

1. Summary
2. Extrapolation to solve ODE
3. Error-free Transformation and its application
4. Numerical Experiments
5. Conclusion and future work

Summary

- For initial value problems of ordinary differential equations (ODEs), we want to obtain more precise double precision numerical solutions more quickly than when using double-double (DD) precision arithmetic.
- We have implemented lighter and accurate BLAS1 functions with EFT and used them to explicit extrapolation methods.

$$\Downarrow$$

The presented routines can be effective for a large system of linear ODE and for small nonlinear ODE, especially when a harmonic sequence is used.

Initial value problem of ordinary differential equation

Initial value problem of Ordinary Differential Equation (ODE for short) to be solved:

$$\begin{aligned} \frac{d\mathbf{y}}{dt} &= \mathbf{f}(t, \mathbf{y}) \in \mathbb{R}^n \\ \mathbf{y}(0) &= \mathbf{y}_0 \end{aligned} \quad . \tag{1}$$

$$\text{Integration interval} : [0, t_{\mathrm{end}}]$$

$$\Downarrow$$

We compute $\mathbf{y}_{\mathrm{next}} \approx \mathbf{y}(t_{\mathrm{next}})$ at each $t_{\mathrm{next}} \in [0, t_{\mathrm{end}}]$ from $\mathbf{y}_{\mathrm{old}} \approx \mathbf{y}(t_{\mathrm{old}})$.

Extrapolation for ODE: Bulirsch-Stoer Algorithm

Give a support sequence $\{w_i\}$, max. number of stages L, relative tolerance ε_R and absolute tolerance ε_A.

Support sequences:

Romberg: $2, 4, 8, ..., 2^i, ... \Rightarrow$ Stable but Slow

Harmonic: $2, 4, 6, 8, ..., 2(i+1), ... \Rightarrow$ Unstable but Fast

Process to calculate initial sequence: $\mathbf{T}_{i1}$ $(i = 1, 2, ..., L)$:

1. $h := (t_{\text{next}} - t_{\text{old}})/w_i \longrightarrow t_k := t_{\text{old}} + kh \in [t_{\text{old}}, t_{\text{next}}]$
2. $t_0 := t_{\text{old}},\ y_0 \approx y(t_0)$
3. Explicit Euler Method

$$\mathbf{y}_1 := \mathbf{y}_0 + h\mathbf{f}(t_0, \mathbf{y}_0)$$

4. Explicit midpoint method to get $\mathbf{y}_2$, $\mathbf{y}_3$, ... $\mathbf{y}_{w_i}$

$$\mathbf{y}_{k+1} := \mathbf{y}_{k-1} + 2h\mathbf{f}(t_k, \mathbf{y}_k)\ (k = 1, 2, ..., w_i - 1)$$

5. Set the initial sequence for extrapolation: $\mathbf{S}(h/w_i) := \mathbf{y}_{w_i}$

Extrapolation for ODE: Bulliursh-Stoer Algorithm (cont.)

1. $\mathbf{T}_{11} := \mathbf{S}(h/w_1)$
2. $i = 2, ..., L$

   $\mathbf{T}_{i1} := \mathbf{S}(h/w_i)$
   For $j = 2, ..., i$

   Extrapolation to get better approximation:

   $$\mathbf{R}_{ij} := \left(\left(\frac{w_i}{w_{i-j+1}}\right)^2 - 1\right)^{-1} (\mathbf{T}_{i,j-1} - \mathbf{T}_{i-1,j-1})$$

   $$\mathbf{T}_{ij} := \mathbf{T}_{i,j-1} + \mathbf{R}_{ij}$$

   Check convergence status if :

   $$\begin{aligned} \|\mathbf{R}_{ij}\| \leq \varepsilon_R \|\mathbf{T}_{i,j-1}\| + \varepsilon_A \\ \longrightarrow \mathbf{y}_{\text{next}} := \mathbf{T}_{ij} \end{aligned} \tag{2}$$

3. $\mathbf{y}_{\text{next}} := \mathbf{T}_{LL}$ if not converge

Application of EFT: FMA with error

cf. S.Boldo & J-M. Muller

$(s,\ e_1,\ e_2) := \mathsf{FMAerror}(a,\ x,\ y)$
$s := \mathsf{FMA}(a, x, y) = ax + y$
$(u_1, u_2) := \mathsf{TwoProd}(a, x)$
$(\alpha_1, \alpha_2) := \mathsf{TwoSum}(y, u_2)$
$(\beta_1, \beta_2) := \mathsf{TwoSum}(u_1, \alpha_1)$
$\gamma := \beta_1 \ominus s \oplus \beta_2$
$(e_1, e_2) := \mathsf{QuickTwoSum}(\gamma, \alpha_2)$
return $(s,\ e_1,\ e_2)$

$$s + e_1 + e_2 = ax + y$$

where $s = a \otimes x \oplus y$

$$|e_1 + e_2| = \frac{1}{2}\mathbf{u}|s| \ (\mathbf{u} \text{ is unit of round-off error})$$

$$|e_2| = \frac{1}{2}\mathbf{u}|e_1|$$

Application of EFT2: FMA with error approximated

cf. S.Boldo & J-M. Muller

$(s,\, e) :=$ FMAerrorApprox$(a,\, x,\, y)$
$s :=$ FMA(a, x, y)
$(u_1, u_2) :=$ TwoProd(a, x)
$(\alpha_1, \alpha_2) :=$ TwoSum(y, u_1)
$\gamma := \alpha_1 \ominus s$
$e := (u_2 \oplus \alpha_2) \oplus \gamma$
return $(s,\, e)$

When IEEE754 double precision arithmetic is used in FMAerrorApprox, the error bound is provided as $|(s+e)-(ax+b)| \leq 7 \cdot 2^{-105}|s|$.

Application of EFT: BLAS1 with error

$\mathbf{y} := \mathsf{AXPY}(\alpha, \mathbf{x}, \mathbf{y})$
$\quad \mathbf{y} := \alpha \otimes \mathbf{x} \oplus \mathbf{y}$
$\quad$ **return** $\mathbf{y}$

$$\Downarrow$$

$(\mathbf{y}, \mathbf{e_y}) := \mathsf{AXPYerror}(\alpha, e_\alpha, \mathbf{x}, \mathbf{e_x}, \mathbf{y}, \mathbf{e_y})$
$\quad (\mathbf{y}, \mathbf{e}_1, \mathbf{e}_2) := \mathsf{FMAerror}(\alpha, \mathbf{x}, \mathbf{y})$
$\quad \mathbf{e_y} := \mathbf{e}_1 \oplus \mathbf{e}_2 \oplus \alpha \otimes \mathbf{e_x} \oplus e_\alpha \otimes \mathbf{x} \oplus \mathbf{e_y}$
$\quad$ **return** $(\mathbf{y},\ \mathbf{e_y})$

or

$(\mathbf{y}, \mathbf{e_y}) := \mathsf{AXPYerrorA}(\alpha, e_\alpha, \mathbf{x}, \mathbf{e_x}, \mathbf{y}, \mathbf{e_y})$
$\quad (\mathbf{y}, \mathbf{e}) := \mathsf{FMAerrorApprox}(\alpha, \mathbf{x}, \mathbf{y})$
$\quad \mathbf{e_y} := \mathbf{e} \oplus \alpha \otimes \mathbf{e_x} \oplus e_\alpha \otimes \mathbf{x} \oplus \mathbf{e_y}$
$\quad$ **return** $(\mathbf{y},\ \mathbf{e_y})$

Application of EFT: BLAS1 with error

$\mathbf{x} := \mathsf{SCAL}(\alpha, \mathbf{x})$
$\quad \mathbf{x} := \alpha \otimes \mathbf{x}$
$\quad$ **return** $\mathbf{x}$

$$\Downarrow$$

$(\mathbf{x}, \mathbf{e_x}) := \mathsf{SCALerror}(\alpha, e_\alpha, \mathbf{x}, \mathbf{e_x})$
$\quad (\mathbf{w}_1, \mathbf{w}_2) := \mathsf{TwoProd}(\alpha, \mathbf{x})$
$\quad \mathbf{w}_2 := \alpha \otimes \mathbf{e_x} \oplus e_\alpha \otimes (\mathbf{x} \oplus \mathbf{e_x}) \oplus \mathbf{w}_2$
$\quad (\mathbf{x}, \mathbf{e_x}) := \mathsf{QuickTwoSum}(\mathbf{w}_1, \mathbf{w}_2)$
$\quad$ **return** $(\mathbf{x}, \mathbf{e_x})$

Extrapolation with EFT

Approximation $\Longrightarrow$(Approximation, its error)

$\mathbf{f}(t_k, \mathbf{y}_k) := \mathbf{f}_k \implies \mathbf{f}(t_k + e_{t_k}, \mathbf{y}_k + \mathbf{e}_{\mathbf{y_k}}) = \mathbf{f}_k + \mathbf{e}_{\mathbf{f}_k}$

Explicit Euler Method

$$\mathbf{y}_1 := \mathbf{y}_0 + h\mathbf{f}_0$$

$$\Downarrow$$

$$\begin{aligned}
(\mathbf{y}_1, \mathbf{e}_{\mathbf{y}_1}) &:= (\mathbf{y}_0, \mathbf{e}_{\mathbf{y}_0}) \\
(\mathbf{y}_1, \mathbf{e}_{\mathbf{y}_1}) &:= \text{AXPYerror}(h, e_h, \mathbf{f}_0, \mathbf{e}_{\mathbf{f}_0}, \mathbf{y}_1, \mathbf{e}_{\mathbf{y}_1}) \\
\text{or } &:= \text{AXPYerrorA}(h, e_h, \mathbf{f}_0, \mathbf{e}_{\mathbf{f}_0}, \mathbf{y}_1, \mathbf{e}_{\mathbf{y}_1})
\end{aligned}$$

Explicit midpoint method

$$\mathbf{y}_{k+1} := \mathbf{y}_{k-1} + 2h\mathbf{f}_k \ (k = 1, 2, ..., w_i - 1)$$

$$\Downarrow$$

$$\begin{aligned}
(\mathbf{y}_{k+1}, \mathbf{e}_{\mathbf{y}_{k+1}}) &:= (\mathbf{y}_{k-1}, \mathbf{e}_{\mathbf{y}_{k-1}}) \\
(\mathbf{y}_{k+1}, \mathbf{e}_{\mathbf{y}_{k+1}}) &:= \text{AXPYerror}(2 \otimes h, 2 \otimes e_h, \mathbf{f}_k, \mathbf{e}_{\mathbf{f}_k}, \mathbf{y}_{k+1}, \mathbf{e}_{\mathbf{y}_{k+1}}) \\
\text{or } &:= \text{AXPYerrorA}(2 \otimes h, 2 \otimes e_h, \mathbf{f}_k, \mathbf{e}_{\mathbf{f}_k}, \mathbf{y}_{k+1}, \mathbf{e}_{\mathbf{y}_{k+1}}) \\
&(k = 1, 2, ..., w_i - 1)
\end{aligned}$$

Extrapolation with EFT (cont.)

Extrapolation Process

Preliminary (DD): $(c_{ij}, e_{c_{ij}}) := 1/((w_i/w_{i-j+1})^2 - 1)$

$$(\mathbf{R}_{ij}, \mathbf{e}_{\mathbf{R}_{ij}}) := (\mathbf{T}_{i,j-1}, \mathbf{e}_{\mathbf{T}_{i,j-1}})$$

$$(\mathbf{T}_{ij}, \mathbf{e}_{\mathbf{T}_{ij}}) := (\mathbf{T}_{i,j-1}, \mathbf{e}_{\mathbf{T}_{i,j-1}})$$

$$(\mathbf{R}_{ij}, \mathbf{e}_{\mathbf{R_{ij}}}) := \text{AXPYerror}(-1, 0, \mathbf{T}_{i-1,j-1}, \mathbf{e}_{\mathbf{T}_{i-1,j-1}}, \mathbf{R}_{ij}, \mathbf{e}_{\mathbf{R}_{ij}})$$

$$\text{or} := \text{AXPYerrorA}(-1, 0, \mathbf{T}_{i-1,j-1}, \mathbf{e}_{\mathbf{T}_{i-1,j-1}}, \mathbf{R}_{ij}, \mathbf{e}_{\mathbf{R}_{ij}})$$

$$(\mathbf{R}_{ij}, \mathbf{e}_{\mathbf{R}_{ij}}) := \text{SCALerror}(c_{ij}, e_{c_{ij}}, \mathbf{R}_{ij}, \mathbf{e}_{\mathbf{R}_{ij}})$$

$$(\mathbf{T}_{ij}, \mathbf{e}_{\mathbf{T}_{ij}}) := \text{AXPYerror}(1, 0, \mathbf{R}_{ij}, \mathbf{e}_{\mathbf{R}_{ij}}, \mathbf{T}_{ij}, \mathbf{e}_{\mathbf{T}_{ij}})$$

$$\text{or} := \text{AXPYerrorA}(1, 0, \mathbf{R}_{ij}, \mathbf{e}_{\mathbf{R}_{ij}}, \mathbf{T}_{ij}, \mathbf{e}_{\mathbf{T}_{ij}})$$

Møller method

The Møller method is proposed to reduce the accumulation of round-off errors incurred during the approximation of IVPs of ODEs and is a type of compensated summation. For the original summation $S_i := S_{i-1} + z_{i-1}$, we compute it as follows:

$$
\begin{aligned}
s_i &:= z_{i-1} \ominus R_{i-1} \ (R_0 = 0) \\
S_i &:= S_{i-1} \oplus s_i \\
r_i &:= S_i \ominus S_{i-1} \\
R_i &:= r_i \ominus s_i.
\end{aligned}
$$

$$\Downarrow$$

$$
\begin{gathered}
s_i := z_{i-1} \oplus R'_{i-1} \ (R'_0 = 0) \\
(S_i, R'_i) := \mathsf{QuickTwoSum}(S_{i-1}, s_i).
\end{gathered} \tag{3}
$$

Computing environment

Ryzen AMD Ryzen 1700 (2.7 GHz), Ubuntu 16.04.5, GCC 5.4.0, QD 2.3.18[7], LAPACK 3.8.0.

Corei7 Intel Core i7-9700K (3.6GHz), Ubuntu 18.04.2, GCC 7.3.0, QD 2.3.20, LAPACK 3.8.0.

Targetted algorithms

Our targets of precision are IEEE754 double precision (Double) and DD provided by the QD library. The targeted algorithms are as follows:

DEFT	Double precision and AXPYerror
DEFTA	Double precision, $\mathbf{f} + \mathbf{e_f}$, and AXPYerrorA
DMøller	Double precision Møller method.

DEFTA means the usage of the FMAerrorA in the entire extrapolation process. For DEFT, DEFTA and DD computations, we used DD precision $\mathbf{f}$.

- To check for convergence (4),

$$\|\mathbf{R}_{ij}\| \leq \varepsilon_R \|\mathbf{T}_{i,j-1}\| + \varepsilon_A \tag{4}$$

  we used $\varepsilon_R = \varepsilon_A = 0$ unless otherwise specified.
- All EFT basic functions were coded as C macros.

Numerical experiments

1.

$$\frac{d}{dt}\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} -y_1 \\ \vdots \\ -ny_n \end{bmatrix} \Longrightarrow \mathbf{y}(t) = \begin{bmatrix} \exp(-t) \\ \vdots \\ \exp(-nt) \end{bmatrix}$$

$$\mathbf{y}(0) = [1\ 1\ \ \cdots\ \ 1]^T, t \in [0, 1/4], n = 2048.$$

2.

$$\frac{d}{dt}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ -\alpha y_1^2 \sin t + 2\alpha y_1 y_2 \cos t \end{bmatrix}$$

$$\mathbf{y}(0) = [1\ \alpha]^T,\ t \in [0, 37]$$

where $\alpha = 0.99999999$. The analytical solution is

$$\mathbf{y}(t) = \begin{bmatrix} 1/(1 - \alpha \sin t) \\ \alpha \cos t/(1 - \alpha \sin t)^2 \end{bmatrix}.$$

Problem 1: Simple Linear ODE

$$\frac{d}{dt}\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} -y_1 \\ -2y_2 \\ \vdots \\ -ny_n \end{bmatrix} \Longrightarrow \mathbf{y}(t) = \begin{bmatrix} \exp(-x) \\ \exp(-2x) \\ \vdots \\ \exp(-nx) \end{bmatrix}$$

$$\mathbf{y}(0) = [1\ 1\ \cdots\ 1]^T, t \in [0, 1/4], n = 2048.$$

Problem 1: Simple Linear ODE

Romberg sequence: $L = 4$ at $t_{\text{end}} = 1/4$

$L = 4$ #steps	Computational time (s) on Ryzen DD	DEFT	DEFTA	Double	DMøller
512	1.79	1.41	0.99	0.2	0.33
1024	3.59	2.81	1.95	0.41	0.67
2048	7.18	5.64	3.82	0.81	1.33
4096	14.4	11.3	7.58	1.62	2.66
#steps	Computational time (s) on Corei7				
512	1.17	0.86	0.73	0.1	0.26
1024	2.33	1.69	1.47	0.21	0.52
2048	4.64	3.39	2.92	0.41	1.04
4096	9.34	6.75	5.87	0.82	2.07
#steps	Max. Relative Error				
512	1.8E-07	1.8E-07	1.8E-07	1.8E-07	1.8E-07
1024	1.2E-10	1.2E-10	1.2E-10	1.2E-10	1.2E-10
2048	9.3E-14	9.3E-14	9.3E-14	1.5E-13	9.4E-14
4096	8.2E-17	4.6E-16	4.6E-16	2.3E-13	4.3E-14

Problem 1: Simple Linear ODE

Harmonic sequence: $L = 6$ at $t_{\text{end}} = 1/4$

$L = 6$ #steps	Computational Time (s) on Ryzen				
	DD	DEFT	DEFTA	Double	DMøller
512	1.87	1.76	1.42	0.28	0.4
1024	3.74	3.53	2.84	0.55	0.81
2048	7.48	6.93	5.58	1.11	1.62
4096	14.9	10.4	8.38	2.22	3.24
#steps	Computational Time (s) on Corei7				
512	1.4	1.04	0.89	0.1	0.26
1024	2.8	2.07	1.78	0.21	0.52
2048	5.6	4.11	3.5	0.41	1.04
4096	11.2	6.17	5.27	0.82	2.07
#steps	Max. Relative Error				
512	4.3E-10	4.3E-10	4.3E-10	4.3E-10	4.3E-10
1024	1.7E-14	2.7E-14	2.7E-14	7.1E-13	6.6E-13
2048	8.4E-19	1.3E-14	1.3E-14	9.2E-13	7.2E-13
4096	4.6E-23	5.5E-15	5.5E-15	1.0E-12	7.6E-13

Problem 2: Resonance problem

We pick up the following resonance problem that is necessary to control step sizes.

$$\frac{d}{dt}\left[\begin{array}{c} y_1 \\ y_2 \end{array}\right] = \left[\begin{array}{c} y_2 \\ -\alpha y_1^2 \sin t + 2\alpha y_1 y_2 \cos t \end{array}\right]$$
$$\mathbf{y}(0) = [1\ \alpha]^T,\ t \in [0, 37]$$

where $\alpha = 0.99999999$. The analytical solution is

$$\left[\begin{array}{c} y_1 \\ y_2 \end{array}\right] = \left[\begin{array}{c} 1/(1-\alpha\sin t) \\ \alpha\cos t/(1-\alpha\sin t)^2 \end{array}\right].$$

The algorithm of step size control is the same one proposed in Murofushi and Nagasaka[4], wherein the current step size is halved if the convergent condition (4) is not satisfied. The maximum stages are $L = 12$ for Romberg sequence and $L = 18$ for harmonic sequence as recommended in [4].

Problem 2: Resonance problem

Computational time and maximum relative errors
at $t_{\mathrm{end}} = 37$ with Romberg seq.

Romberg, $L = 12$	#steps	Ryzen Comp.Time (s)	Corei7	Max.Rel.Err.
Double	84	0.360	0.018	1.0E-01
DEFT	100	0.514	0.401	3.7E-04
DEFTA	100	0.507	0.398	3.7E-04
DMøller	98	0.149	0.094	5.2E-04
DD	213	1.895	1.598	1.1E-17

Problem 2: Resonance problem

Computational time and maximum relative errors
at $t_{\mathrm{end}} = 37$ with harmonic seq.

Harmonic, $L = 18$	#steps	Ryzen Comp.Time (s)	Corei7	Max.Rel.Err.
Double	NC			
DEFT	159	0.0458	0.0383	4.5E-04
DEFTA	159	0.0448	0.0378	4.5E-04
DMøller	NC			
DD($\varepsilon_R = 10^{-16}$)	121	0.136	0.077	3.2E-02
DD($\varepsilon_R = 10^{-18}$)	186	0.158	0.098	6.0E-05
DD($\varepsilon_R = 10^{-30}$)	6455	1.53	1.30	4.6E-13

Conclusion

- DEFTA is approximately 1.6 times faster than DD and 1.2 times faster than DEFT.
- There are no differences between DEFT's and DEFTA's approximations.
- DEFT and DEFTA are effective for resonance problem with harmonic sequence.

References

- T. Ogita, S. M. Rump, and S. Oishi, Accurate sum of and dot product, SIAM Journal of Scientific Computing **26**(2005), 1955–1988.
- Y. Kobayashi and T. Ogita, A fast and efficient algorithm for solving ill-conditioned linear systems, JSIAM Letters **7**(2015), 1–4.
- E. Hairer, S. P. Nørsett and G. Wanner, Solving Ordinary Differential Equations I, Springer-Verlarg, New York, 1996.
- M. Murofushi and H. Nagasaka, The relationship between the round-off errors and Møller's algorithm in the extrapolation method, Annals Num., **1**(1994), 451-458.
- O. Møller, Quasi Double-Precision in Floating Point Addition, BIT **5**(1965), 37-50.
- S. Bold and J. -M. Muller, Exact and Approximated Error of the FMA, IEEE Transactions on Computers, **60**(2011), 157–164.