Extrapolation methods and their applications
Exercises

Michela Redivo-Zaglia

Department of Mathematics "Tullio Levi-Civita"

University of Padova, Italy

Email: michela.redivozaglia@unipd.it

This document contains some exercises that have to be solved by the students needing to pass the exam. The solutions, collected in a zipped file, have to contain a pdf file with comments, answers, results and figures (a kind of report), the functions and scripts written by the students and used for obtaining the solutions.

The students may produce the solutions in group (not more that three students) and have to send them (by mail to michela.redivozaglia@unipd.it) not after October 7, 2018.

1. **Exercises on Aitken's Δ^2 process**.

   (a) Given a sequence $(\mathbf{S_n})$ of real (or complex) numbers slowly converging, one can transform it, without modifying its terms, into a new sequence which, under some assumptions, converges faster to the same limit. One of the most well known sequence transformation is Aitken's Δ^2 process. It can be written in several ways.

   Consider the fixed point sequence $(\mathbf{S_n})$ constructed by

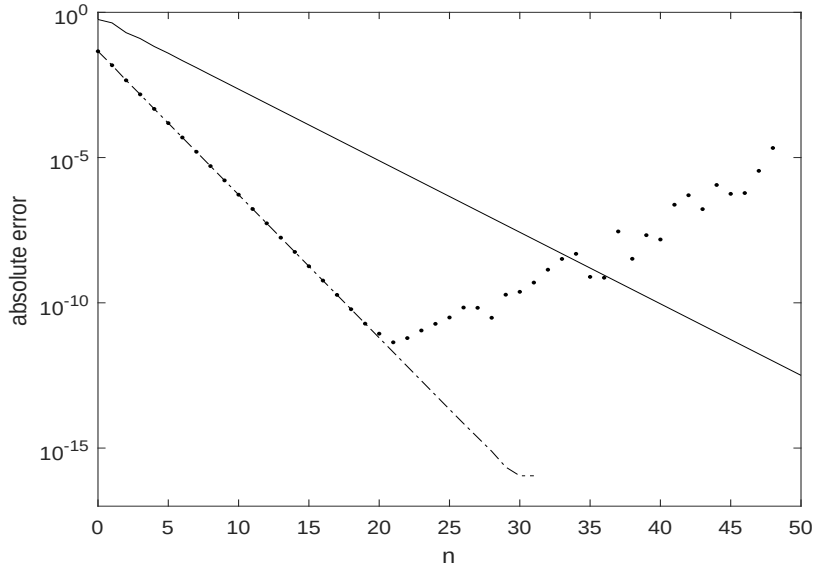
   $$\mathbf{S_{n+1}} = \exp(-\mathbf{S_n}), \quad \mathbf{n = 0, 1, \ldots}, \quad \mathbf{S_0 = 0}.$$

   It converges to $\mathbf{S = 0.5671432904097839\cdots} = \exp(-\mathbf{S})$ (the sign of $\mathbf{S}$ in the slides of Lecture 1 is wrong).

   We apply Aitken's process to it using the two formulas (Δ denote the usual forward difference operator)

   $$\begin{aligned} T_n^1 &= S_n - \frac{(\Delta S_n)^2}{\Delta^2 S_n} \qquad \textbf{stable form} \\ T_n^2 &= \frac{S_n S_{n+2} - S_{n+1}^2}{\Delta^2 S_n} \qquad \textbf{unstable form}. \end{aligned}$$

   Take $n = 0, \ldots, 50$, compute the sequences $(\mathbf{S_n})$, $(\mathbf{T_n^1})$ and $(\mathbf{T_n^2})$ and plot the curves of the absolute errors with respect to the value $\mathbf{S \simeq 0.5671432904097839}$, for obtaining a figure similar to



with (S_n) (plain line), (T_n^1) stable (dash–dotted line) and (T_n^2) unstable (dotted line).

(b) Now, consider the following formulas, all mathematically equivalent the previous ones, and for the same sequence detect if on this example they are stable or not (produce the corresponding figures).

$$
\begin{aligned}
T_n^3 &= S_{n+1} - \frac{\Delta S_n \Delta S_{n+1}}{\Delta^2 S_n} \\
T_n^4 &= \frac{S_n \Delta S_{n+1} - S_{n+1} \Delta S_n}{\Delta^2 S_n} \\
T_n^5 &= S_{n+2} - \frac{(\Delta S_{n+1})^2}{\Delta^2 S_n} \\
T_n^6 &= \frac{S_{n+1} - \dfrac{\Delta S_{n+1}}{\Delta S_n} S_n}{1 - \dfrac{\Delta S_{n+1}}{\Delta S_n}} \\
T_n^7 &= S_{n+1} + \frac{1}{1/\Delta S_{n+1} - 1/\Delta S_n}
\end{aligned}
$$

(c) **Optional. Not easy to do!** Find a scalar sequence where at least one other formula (T_n^2 apart) is unstable.

(d) *The effects of the arithmetic of the computers.* We saw that

$$
T_n^2 = \frac{S_n S_{n+2} - S_{n+1}^2}{S_{n+2} - 2S_{n+1} + S_n}
$$

is in most cases unstable. Try on the example of item (a) the following slight modification of that formula (the only change is the order of the terms of the denominator) and produce a figure comparing the errors

$$
T_n^8 = \frac{S_n S_{n+2} - S_{n+1}^2}{S_{n+2} + S_n - 2S_{n+1}}
$$

The curves coincide? Why?

2. **Exercises on the scalar ε-algorithm**.

(a) The rule of the scalar ε-algorithm, implementing the Shank's transformation, are

$$
\left\{
\begin{array}{lcll}
\varepsilon_{-1}^{(n)} &=& 0, & n = 0, 1, \ldots, \\
\varepsilon_{0}^{(n)} &=& S_n, & n = 0, 1, \ldots, \\
\varepsilon_{k+1}^{(n)} &=& \varepsilon_{k-1}^{(n+1)} + (\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)})^{-1}, & k, n = 0, 1, \ldots
\end{array}
\right. \tag{1}
$$

and we have

$$
\varepsilon_{2k}^{(n)} = e_k(S_n).
$$

The ε's are put in the two–dimensional array (we set $n = 0$), called the ε-array.

If we want to compute the quantity $\varepsilon_{2k}^{(n)}$, $2k + 1$ terms of the original sequence are needed.

For instance, if we fix $2k = 4$ (the column -1 is omitted in the figure), we have

	column 0	column 1	column 2	column 3	column 4
diagonal 1	$\varepsilon_0^{(0)} = S_0$				
		$\varepsilon_1^{(0)}$			
diagonal 2	$\varepsilon_0^{(1)} = S_1$		$\varepsilon_2^{(0)}$		
		$\varepsilon_1^{(1)}$		$\varepsilon_3^{(0)}$	
diagonal 3	$\varepsilon_0^{(2)} = S_2$		$\varepsilon_2^{(1)}$		$\varepsilon_4^{(0)}$
		$\varepsilon_1^{(2)}$		$\varepsilon_3^{(1)}$	
diagonal 4	$\varepsilon_0^{(3)} = S_3$		$\varepsilon_2^{(2)}$		$\varepsilon_4^{(1)}$
		$\varepsilon_1^{(3)}$		$\varepsilon_3^{(2)}$	
diagonal 5	$\varepsilon_0^{(4)} = S_4$		$\varepsilon_2^{(3)}$		$\varepsilon_4^{(2)}$
		$\varepsilon_1^{(4)}$		$\varepsilon_3^{(3)}$	
diagonal 6	$\varepsilon_0^{(5)} = S_5$		$\varepsilon_2^{(4)}$		
		$\varepsilon_1^{(5)}$			
diagonal 7	$\varepsilon_0^{(6)} = S_6$				
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Implement the ε–algorithm **by columns**, storing the whole two-dimensional array. The first row of the array will contain the first descending diagonal and so on.

We know that if we consider a formal power series

$$f(t) = c_0 + c_1 t + c_2 t^2 + \cdots$$

the Padé approximant of such series, usually denoted by $[p/q]_f(t)$, is linked to the ε–algorithm. In fact, applying the ε–algorithm to the partial sums (S_n) of f, then

$$e_k(S_n) = \varepsilon_{2k}^{(n)} = [n + k/k]_f(t).$$

We consider the series

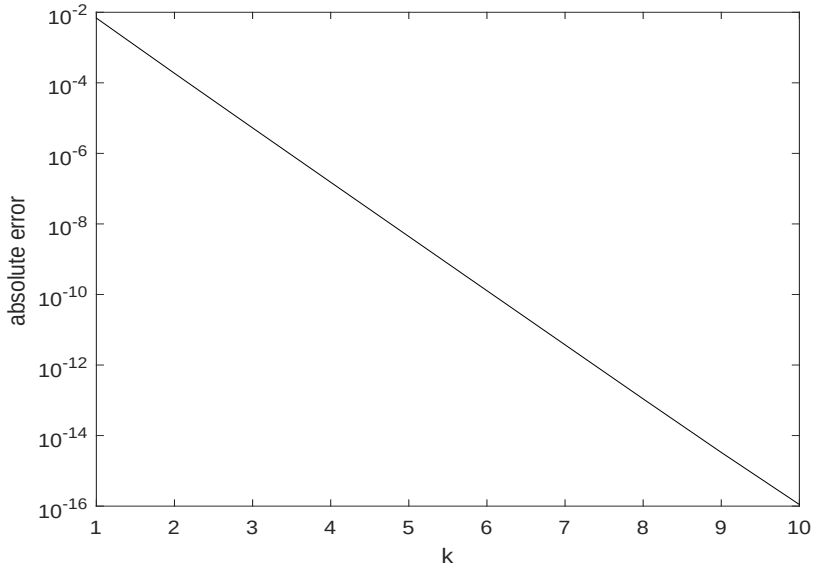
$$f(t) = t - t^2/2 + t^3/3 - t^4/4 + \cdots$$

which converge to $\ln(1 + t)$ for $|t| \leq 1, t \neq -1$.

For $t = 1$, we have $\ln 2 = 0.6931471805599453\ldots$

Apply the ε–algorithm to the sequence of the partial sums $S_0, S_1, S_2, \ldots, S_{21}$ and compute $\varepsilon_2^{(0)} = [1/1]_f(1), \varepsilon_4^{(0)} = [2/2]_f(1), \ldots, \varepsilon_{20}^{(0)} = [10/10]_f(1)$.

Produce a figure of the absolute errors. The result has to be similar to



3. **Not easy to do!** Implementing the ε–algorithm **by diagonals**, with the moving lozenge technique of Wynn. It consists in storing the last ascending diagonal of the ε–array (in this diagonal the sum of the lower and the upper indexes is constant), that is, for example, $\varepsilon_0^{(m)} = S_m, \varepsilon_1^{(m-1)}, \ldots, \varepsilon_m^{(0)}$, and to add, one by one (that is $\varepsilon_0^{(m+1)} = S_{m+1}$), the terms of the sequence to be transformed. Then, a new ascending diagonal is built step-by-step, by moving up the lozenge, and the new diagonal gradually replaces the old one. In this way, the algorithm needs to store only one vector and three auxiliary variables (see the notes distributed, and the original paper of Wynn).

   Apply the new implementation to the same example given before and recover the results.

4. **Optional.** Compare the numerical results obtained by using the EPSfun Matlab toolbox (function `VEAW`) freely available in `netlib` (package `na44` of `numeralgo`). In the `demo` directory there is a script file that can help the students in this work. The function is general (that is it can be used also for vectors) but it works also for scalar sequences.