EPSfun: a Matlab toolbox for implementing the simplified topological $\varepsilon$–algorithms and its applications

Michela Redivo–Zaglia ¹  Claude Brezinski ²

¹Dipartimento di Matematica “Tullio Levi-Civita”, Università degli Studi di Padova, Italy

²Laboratoire Paul Painlevé, Université de Lille 1 - Sciences et Technologies, France

2 giorni di Algebra Lineare Numerica (Como, Italy)
February 16-17, 2017
The scalar Shanks transformation and the $\varepsilon$-algorithm

Particular rules

The topological Shanks transformations and the (simplified) topological $\varepsilon$-algorithms

The EPSFUN Matlab toolbox

Applications and numerical results
Shanks transformation and the \( \varepsilon \)–algorithm

Let \((S_n)\) be a sequence of elements of a vector space \( E \) on a field \( K \) (\( \mathbb{R} \) or \( \mathbb{C} \)) which converges to a limit \( S \).

If the convergence is slow, it can be transformed, by a sequence transformation, into a new sequence (or into a set of new sequences) which, under some assumptions, converges faster to the same limit.

When \( E \) is \( \mathbb{R} \) or \( \mathbb{C} \), a well known such transformation is due to Shanks (1955), and it can be implemented by the scalar \( \varepsilon \)–algorithm of Wynn (1956).

We assume that the sequence \((S_n)\) satisfies, for a fixed value of \( k \), the difference equation

\[
a_0(S_n - S) + \cdots + a_k(S_{n+k} - S) = 0 \in E, \quad n = 0, 1, \ldots
\]

with \( a_0a_k \neq 0 \) and \( a_0 + \cdots + a_k \neq 0 \).
Shanks proposed a transformation able to define a new sequence \((e_k(S_n))\) such that

\[ e_k(S_n) = S, \quad \text{for all } n. \]

This linear combination can be computed even if \((S_n)\) does not satisfy the difference equation and, after having computed the coefficients \(a_0, \ldots, a_k\) (now depending on \(k\) and \(n\)) and by adding a normalization condition \(a_0 + \cdots + a_k = 1\), we set

\[ e_k(S_n) = a_0 S_n + \cdots + a_k S_{n+k}. \]
For computing recursively the $e_k(S_n)$ we can use the Wynn scalar $\varepsilon$–algorithm whose rule are

\[
\begin{align*}
\varepsilon_{n-1}^{(n)} &= 0, \quad n = 0, 1, \ldots, \\
\varepsilon_0^{(n)} &= S_n, \quad n = 0, 1, \ldots, \\
\varepsilon_k^{(n)} &= \varepsilon_{k-1}^{(n+1)} + (\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)})^{-1}, \quad k, n = 0, 1, \ldots
\end{align*}
\]

and it holds

\[
\varepsilon_{2k}^{(n)} = e_k(S_n), \quad \varepsilon_{2k+1}^{(n)} = 1/e_k(\Delta S_n)
\]

Thus only the quantities with an even lower index are of interest (the quantities with an odd lower index are intermediate computations).
The $\epsilon$-array

The $\epsilon$’s are put in a table, called the $\epsilon$-array

\[
\begin{align*}
\epsilon^{(0)}_{-1} &= 0 \\
\epsilon^{(0)} &= S_0 \\
\epsilon^{(1)}_{-1} &= 0 \\
\epsilon^{(1)} &= S_1 \\
\epsilon^{(2)}_{-1} &= 0 \\
\epsilon^{(2)} &= S_2 \\
\epsilon^{(3)}_{-1} &= 0 \\
\epsilon^{(3)} &= S_3 \\
\epsilon^{(4)}_{-1} &= 0 \\
\epsilon^{(4)} &= S_4 \\
\end{align*}
\]
The quantities related by the $\varepsilon$-algorithm give the so called rhombus scheme, were the quantity in red is computed from the three ones in blue.

![Diagram of rhombus scheme]

The algorithm is implemented by ascending diagonals and only one vector and three auxiliary variables are needed.
The $\varepsilon$-array with $\text{MAXCOL}=4$

<table>
<thead>
<tr>
<th>diagonal 1</th>
<th>diagonal 2</th>
<th>diagonal 3</th>
<th>diagonal 4</th>
<th>diagonal 5</th>
<th>diagonal 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon_0^{(0)} = S_0$</td>
<td>$\varepsilon_0^{(1)} = S_1$</td>
<td>$\varepsilon_0^{(2)} = S_2$</td>
<td>$\varepsilon_0^{(3)} = S_3$</td>
<td>$\varepsilon_0^{(4)} = S_4$</td>
<td>$\varepsilon_0^{(5)} = S_5$</td>
</tr>
<tr>
<td>$\varepsilon_1^{(0)}$</td>
<td>$\varepsilon_1^{(1)}$</td>
<td>$\varepsilon_1^{(2)}$</td>
<td>$\varepsilon_1^{(3)}$</td>
<td>$\varepsilon_1^{(4)}$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\varepsilon_2^{(0)}$</td>
<td>$\varepsilon_2^{(1)}$</td>
<td>$\varepsilon_2^{(2)}$</td>
<td>$\varepsilon_2^{(3)}$</td>
<td>$\varepsilon_2^{(4)}$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\varepsilon_3^{(0)}$</td>
<td>$\varepsilon_3^{(1)}$</td>
<td>$\varepsilon_3^{(2)}$</td>
<td>$\varepsilon_3^{(3)}$</td>
<td>$\varepsilon_3^{(4)}$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\varepsilon_4^{(0)}$</td>
<td>$\varepsilon_4^{(1)}$</td>
<td>$\varepsilon_4^{(2)}$</td>
<td>$\varepsilon_4^{(3)}$</td>
<td>$\varepsilon_4^{(4)}$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\varepsilon_5^{(0)}$</td>
<td>$\varepsilon_5^{(1)}$</td>
<td>$\varepsilon_5^{(2)}$</td>
<td>$\varepsilon_5^{(3)}$</td>
<td>$\varepsilon_5^{(4)}$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
</tbody>
</table>
Particular rules

\[ \varepsilon_{k-3}^{(n+2)} \quad \varepsilon_{k-2}^{(n+1)} \quad \varepsilon_{k-1}^{(n)} \quad \varepsilon_k \quad \varepsilon_{k+1}^{(n)} \quad \varepsilon_{k+2}^{(n+1)} \quad \varepsilon_{k+3}^{(n+2)} \]

Instead of computing \( E \) as

\[ E = C + 1/(d - b) \]

we use

\[ E = r(1 + r/C)^{-1} \]

with

\[ r = S(1 - S/C)^{-1} + N(1 - N/C)^{-1} - W(1 - W/C)^{-1} \]
**Particular rules: an example of isolated singularity**

| diagonal 1 | \( \varepsilon_0^{(0)} = 0 \) | \( \varepsilon_0^{(1)} = 1 \) | \( \varepsilon_0^{(2)} = 3 \) | \( \varepsilon_0^{(3)} = 5 \) |
| diagonal 2 | \( \varepsilon_0^{(0)} = 1 \) | \( \varepsilon_0^{(1)} = \frac{1}{2} \) | \( \varepsilon_0^{(2)} = \frac{1}{2} \) | \( \varepsilon_0^{(3)} = \frac{3}{4} \) |
| diagonal 3 | \( \varepsilon_0^{(0)} = -1 \) | \( \varepsilon_0^{(1)} = \frac{1}{2} \) | \( \varepsilon_0^{(2)} = \infty \) | \( \varepsilon_0^{(3)} = \frac{1}{2} \) |
| diagonal 4 | \( \varepsilon_0^{(0)} = \frac{1}{2} \) | \( \varepsilon_0^{(1)} = \frac{1}{2} \) | \( \varepsilon_0^{(2)} = 9 \) | \( \varepsilon_0^{(3)} = \frac{1}{2} \) |
| diagonal 5 | \( \varepsilon_0^{(0)} = \frac{19}{3} \) | \( \varepsilon_0^{(1)} = 1 \) | \( \varepsilon_0^{(2)} = 9 \) | \( \varepsilon_0^{(3)} = \frac{1}{2} \) |
Let us consider the case where $E$ is a general vector space, and we assume again that the elements of the sequence $S_n \in E$ satisfy the difference equation

$$a_0(S_n - S) + \cdots + a_k(S_{n+k} - S) = 0 \in E, \quad n = 0, 1, \ldots$$

Brezinski (1975) proposed new transformations called **topological $\varepsilon$-transformations**.
For computing the coefficients $a_0, \ldots, a_k$ he considered the relation

$$a_0 \Delta S_n + \cdots + a_k \Delta S_{n+k} = 0.$$

Given $y$, an element of the dual space $E^*$ of $E$ (which means that it is a **linear functional**), writing the relation from $n$ to $n + k - 1$, taking the duality product with $y$, and adding the normalization condition he obtained a **system of scalar relations**.
Once the unknown coefficients have been computed, he defined the **first topological Shanks transformation** by

\[
e_k(S_n) = a_0 S_n + \cdots + a_k S_{n+k},
\]

and the **second topological Shanks transformation** by

\[
\tilde{e}_k(S_n) = a_0 S_{n+k} + \cdots + a_k S_{n+2k}.
\]

By construction, \( \forall n, e_k(S_n) = \tilde{e}_k(S_n) = S \) if

\[
\forall n, a_0(S_n - S) + \cdots + a_k(S_{n+k} - S) = 0.
\]
Each of these transformations can be implemented by recursive algorithms named the Topological $\varepsilon$–algorithms (TEA). However, these algorithms are quite complicated:

- they possess two rules,
- they require the storage of elements of $E$ and of $E^*$,
- the duality product with $y$ is recursively used in their rules.
Each of these transformations can be implemented by recursive algorithms named the **Topological $\varepsilon$–algorithms (TEA)**. However, these algorithms are quite complicated:

- they possess **two rules**,  
- they require the **storage of elements of $E$ and of $E^*$**,  
- the duality product with $y$ is recursively used in their rules.

Recently, simplified versions were obtained and called the **Simplified Topological $\varepsilon$–algorithms (STEA)**.

- **only one recursive rule**,  
- they require **less storage** than the initial algorithms and **only elements of $E$**,  
- the **elements of the dual vector space $E^*$** no longer have to be used in the recursive rules (**only in their initializations**),  
- **numerical stability is improved** (thanks to particular rules of Wynn).
The rule of the First Simplified Topological $\varepsilon$–algorithm, denoted by $\text{STEA1}$, is

$$
\varepsilon_{2k+2}^{(n)} = \varepsilon_{2k}^{(n+1)} + \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n)} - \varepsilon_{2k}^{(n)}} \left(\varepsilon_{2k}^{(n+1)} - \varepsilon_{2k}^{(n)}\right), \quad k, n = 0, 1, \ldots,
$$

with $\varepsilon_0^{(n)} = S_n \in E, \ n = 0, 1, \ldots$

The scalars $\varepsilon_{2k}^{(n)}$ are computed by the scalar $\varepsilon$–algorithm of Wynn, by setting the initial values

$$
\varepsilon_0^{(n)} = \langle y, S_n \rangle, \quad n = 0, 1, \ldots
$$

A Second Simplified Topological $\varepsilon$–algorithm, denoted by $\text{STEA2}$, also exists and the quantities are denoted $\tilde{\varepsilon}_{2k}^{(n)}$.

It holds

$$
\varepsilon_{2k}^{(n)} = e_k(S_n) \quad \tilde{\varepsilon}_{2k}^{(n)} = \tilde{e}_k(S_n).
$$
TEA1 vs STEA1

Odd rule TEA1

Even rule TEA1

STEA1

\[ \varepsilon(n)_{2k} \]

\[ \varepsilon(n)_{2k+1} \]

\[ \varepsilon(n+1)_{2k} \]

\[ \varepsilon(n+1)_{2k+1} \]

\[ \varepsilon(n+1)_{2k-1} \]

\[ \varepsilon(n+1)_{2k+2} \]

\[ \varepsilon(n+1)_{2k} \]

\[ \varepsilon(n)_{2k} \]

\[ \varepsilon(n+1)_{2k+1} \]

\[ \varepsilon(n+1)_{2k+2} \]
Exploitation of the algorithms

The algorithms will be used in two different ways

- **Acceleration method (AM)**
- **Restarted method (RM)**
The algorithms will be used in two different ways

- **Acceleration method (AM)**
- **Restarted method (RM)**

The **Acceleration Method (AM)** can be applied to the solution of systems of linear and nonlinear equations, to the computation of matrix functions, . . . .

Choose 2\(k\) and \(x_0\).

For \(n = 1, 2, \ldots\)

- Compute \(x_n\).
- Apply the STEA1 to \(x_0, x_1, x_2, \ldots\) and compute the sequence of extrapolated values
  \[
  \varepsilon_0^{(0)} = x_0, \varepsilon_0^{(1)}, \varepsilon_2^{(0)}, \varepsilon_2^{(1)}, \ldots, \varepsilon_{2k}^{(0)}, \varepsilon_{2k}^{(1)}, \varepsilon_{2k}^{(2)}, \ldots
  \]

end
The **Restarted Method (RM)** is used for **fixed point problems** in which we have to find the solution of linear or nonlinear equations of the form

\[
x = F(x) \quad \text{or} \quad f(x) = F(x) - x = 0 \quad \text{or} \quad x = x + \alpha f(x),
\]

with \( F : \mathbb{R}^m \mapsto \mathbb{R}^m \).

Choose 2\(k\) and \(x_0\).
For \(i = 0, 1, \ldots\) (cycles or outer iterations)
  Set \(u_0 = x_i\)
  For \(n = 1, \ldots, 2k\) (basic or inner iterations)
    Compute \(u_n = F(u_{n-1})\)
    Apply STEA1 to \(u_0, \ldots, u_{2k}\)
end
Set \(x_{i+1} = \varepsilon^{(0)}_{2k}\)
end
When \( k = m \) (the dimension of the system) the RM is called Generalized Steffensen Method (GSM) and, under some assumptions, the sequence \((x_i)\) of the vertices of the successive \(\varepsilon\)-arrays asymptotically \textit{converges quadratically} to the fixed point \(x\) of \(F\).

**Remark:** This method can also be applied when \(F : \mathbb{R}^{m \times s} \rightarrow \mathbb{R}^{m \times s}\) but the quadratic convergence of the GSM has not yet been proved in this case.


A Matlab toolbox called EPSfun will be published with the paper.

EPSfun Matlab toolbox, na44 package in Netlib (www.netlib.org/numeralgo/).


A Matlab toolbox called **EPSfun** will be published with the paper.

**EPSfun** Matlab toolbox, **na44 package** in *Netlib* (www.netlib.org/numeralgo/).

The EPSFUN Matlab toolbox

- **STEAFun** This directory contains all the STEA functions for implementing the simplified topological $\varepsilon$-algorithms and the SEAW function for the scalar $\varepsilon$-algorithm with particular rules of Wynn.

- **templates** This directory contains some examples of simple scripts for implementing the AM and RM methods.

- **demo** This directory contains demo scripts able to replicate all the examples of the paper.

- **TEAEfun** This directory contains the functions for implementing the original topological $\varepsilon$-algorithms TEA and the vector $\varepsilon$-algorithm of Wynn VEAW (for allowing possible comparisons).

- A detailed userguide (27 pages)
Pseudocode AM

Initializations
Set $\text{MAXCOL}$, $\text{TOL}$, $\text{NDIGIT}$, $\text{NBC}$
Set $\mathbf{y}$

Computations
$\text{IFLAG} \leftarrow 0$
$\text{EPSINI} \leftarrow \mathbf{S}_0$
$\text{EPSINIS} \leftarrow \mathbf{S}_0 = \langle \mathbf{y}, \mathbf{S}_0 \rangle$
First call of SEAW
First call of STEA
for $n = 1 : \text{NBC} - 1$
\quad $\text{EPSINI} \leftarrow \mathbf{S}_n$
\quad $\text{EPSINIS} \leftarrow \mathbf{S}_n = \langle \mathbf{y}, \mathbf{S}_n \rangle$
\quad Call SEAW
\quad Call STEA
\quad Output $\text{EPSINI}$
end for $n$
We suppose to have chosen a certain $\text{MAXCOL} = 2k$. We have (without the storage of the original sequence)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># elements $\varepsilon$-array</th>
<th>In spaces</th>
<th># auxiliary elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEA1</td>
<td>$3k$</td>
<td>$E, E^*$</td>
<td>3</td>
</tr>
<tr>
<td>TEA2</td>
<td>$2k$</td>
<td>$E, E^*$</td>
<td>3</td>
</tr>
<tr>
<td>STEA1</td>
<td>$2k$</td>
<td>$E$</td>
<td>2</td>
</tr>
<tr>
<td>STEA2</td>
<td>$k$</td>
<td>$E$</td>
<td>2</td>
</tr>
</tbody>
</table>
We consider the nonlinear system $f(x) = 0$ of dimension $m = 10$,

$$
  f_i(x) = m - \sum_{j=1}^{m} \cos x_j + i(1 - \cos x_i) - \sin x_i, \quad i = 1, \ldots, m,
$$

whose solution is zero.

We take $x_0 = (1/(2n), \ldots, 1/(2n))^T$, the basic method $x = x + \alpha f(x)$ with $\alpha = 0.1$, $k = 2$ for the AM and $k = 10$ for the RM.

We obtain the results of the next Figure. They show that instability occurs for the AM after iteration 75 (where the error attains $2.7 \cdot 10^{-11}$), and that the GSM achieved a much better precision with a fewer number of iterations (after 2 cycles the STEA1 has an error of $4.62 \cdot 10^{-14}$, and the STEA2 an error of $3.85 \cdot 10^{-14}$).
EXAMPLE 2. AM with STEA1 and STEA2

\[ \|x - \text{sol}\|, \|\text{eps} - \text{sol}\|_{\text{stea1}}, \|x - \text{sol}\|_{\text{stea2}}, \|\text{eps} - \text{sol}\|_{\text{stea2}} \]

EXAMPLE 2. GSM with STEA1 and STEA2

\[ \|x_{\text{ori}} - \text{sol}\|, \|x - \text{sol}\|_{\text{stea1}}, \|\text{eps} - \text{sol}\|_{\text{stea1}}, \|x - \text{sol}\|_{\text{stea2}}, \|\text{eps} - \text{sol}\|_{\text{stea2}} \]

Acceleration Method

Generalized Steffensen Method
We consider the nonlinear system

\[
\begin{align*}
    x_1 &= x_1 x_2^3 / 2 - 1/2 + \sin x_3 \\
    x_2 &= (\exp(1 + x_1 x_2) + 1)/2 \\
    x_3 &= 1 - \cos x_3 + x_1^4 - x_2,
\end{align*}
\]

whose solution is \((-1, 1, 0)^T\).

Starting from \(x_0 = 0\), we obtain the results of the next Figure (left) for the AM with \(\alpha = 0.2\) and \(k = 4\).

For the GSM with \(\alpha = 0.1\) and \(k = 3\) the STEA2 gives better results than the STEA1 as shown on the Figure (right).
EXAMPLE 3. AM with STEA1 and STEA2

EXAMPLE 3. GSM with STEA1 and STEA2

Acceleration Method

Generalized Steffensen Method
Consider now the non–differentiable system

\[
\begin{align*}
|x_1^2 - 1| + x_2 - 1 &= 0 \\
x_2^2 + x_1 - 2 &= 0.
\end{align*}
\]

It has **two solutions**: 
(1, 1)\(^T\) for which we are starting from \(x_0 = (1.3, 1.3)^T\), with \(\alpha = -0.1\) (First Figure - left), 
and \((-2, -2)\) for which we are starting from \(x_0 = (-1, -1)^T\), with \(\alpha = 0.1\) (First Figure - right).

For the **AM** we took \(k = 3\). We see that, for both solutions, the AM works quite well. 
The **GSM** with \(k = 2\) achieves a precision of \(10^{-15}\) in three cycles.
EXAMPLE 4a. AM with STEA1 and STEA2

EXAMPLE 4b. AM with STEA1 and STEA2

AM, first solution

AM, second solution
EXAMPLE 4a. GSM with STEA1 and STEA2

EXAMPLE 4b. GSM with STEA1 and STEA2

GSM, first solution

GSM, second solution
For the nonlinear system

\[ \sum_{j=1}^{7} x_j - (x_i + e^{-x_i}) = 0, \quad i = 1, \ldots, 7, \]

the solution is the vector with all components equal to \(0.14427495072088622350\).

Starting from \(x_0 = (1, \ldots, 1)^T/10\), \(\alpha = -0.01\) and with \(k = 3\) for the AM, we obtain the results of the following Figure.

Notice that with GSM with \(k = 7\) achieves with full precision in only two iterations.
EXAMPLE 5. AM with STEA1 and STEA2

EXAMPLE 5. GSM with STEA1 and STEA2

Acceleration Method

Generalized Steffensen Method
We consider the linear system $AX = B$, where $A$ is the parter matrix of dimension 5 divided by 3 (its spectral radius is 0.9054 and its condition number is 2.149), $X$ is formed by the first two columns of the matrix $pei$ and $B$ is computed accordingly.

We perform the iterations

$$X_{n+1} = (I - A)X_n + B$$

starting from $X_0 = 0$.

The linear functional $y$ associates to the matrix the sum of its elements.

We obtain for the AM the results of next Figure, with $k = 3$ (left) and $k = 4$ (right).

For $k = 5$ (that is for column 10 of the $\varepsilon$–array), the solution is obtained with full precision in one iteration by the GSM as stated by the theory.
EXAMPLE 6. AM with STEA1 and STEA2, \( k=3 \)

\[ \|x - \text{sol}\|, \|\epsilon - \text{sol}\|_{\text{stea1}}, \|\epsilon - \text{sol}\|_{\text{stea2}} \]

AM, \( k = 3 \)

EXAMPLE 6. AM with STEA1 and STEA2, \( k=4 \)

\[ \|x - \text{sol}\|, \|\epsilon - \text{sol}\|_{\text{stea1}}, \|\epsilon - \text{sol}\|_{\text{stea2}} \]

AM, \( k = 4 \)
We now want to solve the matrix equation

\[ X = \sum_{i=0}^{\infty} A_i X^i. \]

We use the algorithm of Z.-Z. Bai (1997) which consists, for \( n = 1, 2, \ldots \), in the iterations

\[
\begin{align*}
Q_n &= I - \sum_{i=1}^{\infty} A_i X_{n-1}^{i-1}, \\
B_n &= 2B_{n-1} - B_{n-1}Q_nB_{n-1}, \\
X_n &= B_nA_0,
\end{align*}
\]

starting from a given \( A_0, B_0 = I \) and \( X_0 = B_0A_0 \). He proved that the sequence \( (X_n) \) converges to the minimal nonnegative solution of the matrix equation.
We consider the numerical example treated by C.-H. Guo (1999), were

\[
A_0 = \frac{4}{3}(1 - p) \begin{pmatrix}
0.05 & 0.1 & 0.2 & 0.3 & 0.1 \\
0.2 & 0.05 & 0.1 & 0.1 & 0.3 \\
0.1 & 0.2 & 0.3 & 0.05 & 0.1 \\
0.1 & 0.05 & 0.2 & 0.1 & 0.3 \\
0.3 & 0.1 & 0.1 & 0.2 & 0.05 \\
\end{pmatrix},
\]

\[A_i = p^i A_0 \quad \text{for} \quad i = 1, 2, \ldots, \quad \text{and} \quad p = 0.49.\]

The linear functional \( y \) used in the duality product corresponds to the **trace of the matrix**.

The infinite sum in the computation of \( Q_n \) was stopped after **100** terms.

The results are given in the next Figure with \( k = 3 \) for the **AM**, and \( k = 5 \) for the **GSM**.
EXAMPLE 8a. AM with STEA1 and STEA2

EXAMPLE 8a. GSM with STEA1 and STEA2

Acceleration Method

Generalized Steffensen Method
We want now to compute an approximation of

$$\log(I + A) = A - \frac{A^2}{2} + \frac{A^3}{3} - \frac{A^4}{4} + \cdots + (-1)^{n+1} \frac{A^n}{n} + \cdots$$

where $A$ is a real matrix.

We know that the series is convergent if $\rho(A) < 1$.

We will show that the topological $\varepsilon$-algorithm is able to accelerate the convergence of the partial sums of this series and also . . . to converge to an approximation of the value, when the series is divergent!

We consider a random matrix $B$ of dimension $N = 50$, we choose a value $r$, and we define $A = r \times B / \rho(B)$.

We obtain for the AM the results of the next Figure (on the left $r = 0.9, k = 4$, and on the right $r = 1.2, k = 4$). The linear functional $y$ is the trace of a matrix.
AM, $r = 0.9, k = 4$

AM, $r = 1.2, k = 4$
The binomial iteration for computing the square root of $I - C$, where $\rho(C) < 1$, consists in the iterations

$$X_{n+1} = \frac{1}{2}(C + X_n^2), \quad k = 0, 1, \ldots, \quad X_0 = 0.$$ 

The sequence $(X_n)$ converges linearly to $X = I - (I - C)^{1/2}$ and $X_n$ reproduces the series

$$(I - C)^{1/2} = \sum_{i=0}^{\infty} \binom{1/2}{i} (-C)^i = I - \sum_{i=1}^{\infty} \alpha_i C_i, \quad \alpha_i > 0,$$

up to and including the term $C^n$.

For $C$, we took the matrix *moler* of dimension 500 divided by $1.1 \times 10^5$ so that $\rho(C) = 0.9855$. The *AM* gives the results of the following Figure with $k = 2$ on the left and $k = 4$ on the right, for the acceleration of the sequence $(X_n)$. 
EXAMPLE 12. AM with STEA1 and STEA2, $k=2$

\[
\|x - \text{sol}\|, \quad \|\epsilon - \text{sol}\|_{\text{stea1}} \quad \|\epsilon - \text{sol}\|_{\text{stea2}}
\]

EXAMPLE 12. AM with STEA1 and STEA2, $k=4$

\[
\|x - \text{sol}\|, \quad \|\epsilon - \text{sol}\|_{\text{stea1}} \quad \|\epsilon - \text{sol}\|_{\text{stea2}}
\]

AM, $k = 2$

AM, $k = 4$
Tensor $\ell^p$-eigenvalues (with F. Tudisco)

Let us consider a square tensor $t = (t_{i_1,...,i_m}) \in \mathbb{R}^{n \times \cdots \times n}$ and the map $T: \mathbb{R}^n \to \mathbb{R}^n$ defined by

$$T(x)_{i_1} = \sum_{i_2,...,i_m=1}^n t_{i_1,...,i_m}x_{i_2} \cdots x_{i_m}, \quad \text{for } i_1 = 1, \ldots, n$$

where $m$ is the order (or modes) of a tensor (that is the number of dimensions).

A vector $v \in \mathbb{R}^n$ is an $\ell^p$-eigenvector of $t$ with eigenvalue $\lambda \in \mathbb{R}$ if

$$T(v) = \lambda |v|^{p-1} \text{sign}(v) \quad \text{(entry-wise operations)}.$$  

Maximal eigenpair

An $\ell^p$-eigenpair $(\lambda, v)$ is maximal if

$$\lambda = \frac{v^T T(v)}{\|v\|^p} = \max_{x \neq 0} \frac{x^T T(x)}{\|x\|^p}$$
Theorem (Perron-Frobenius)

If \( p > m \), \( t_{i_1},...,i_m \geq 0 \) and is “irreducible” then a maximal eigenpair exists, it is unique and positive.

We want to evaluate the maximal eigenvalue, and we consider as basic method the **Shifted (non-linear) Power-Method**:

\[
v_{k+1} = (T(v_k) + s|v_k|^{p-1}\text{sign}(v_k))^{p'-1}, \quad 1/p + 1/p' = 1, \quad s > 0,
\]

\[
\lambda_{k+1} = \frac{v_{k+1}^T T(v_{k+1})}{\|v_{k+1}\|_p}.
\]

We know that, if \( p > m \), this method converges \( \forall s \in [0, 1) \) (not necessarily to the maximal eigenpair).

But, if \( p > m \), and the tensor is non negative and irreducible, this method converges \( \forall s \in [0, 1) \) to the maximal eigenpair.
Two examples for tensor $\ell^p$-eigenvalues

**First example:** $t \in \mathbb{R}^{3 \times 3 \times 3 \times 3}$ symmetric real tensor (Kofidis, Regalia, SIMAX, 2002)

In the scalar $\varepsilon$-algorithm SEAW we consider $k = 2$
Two examples for tensor $\ell^p$-eigenvalues

**Second example:** $\mathbf{t} = \mathbf{t}(\alpha) \in \mathbb{R}^{3 \times 3 \times 3}$ multi-linear Pagerank tensor (nonnegative and irreducible), $0 < \alpha < 1$. **Larger is $\alpha$, better is the model, but more difficult** (Gleich, Lim, Yu, SIMAX, 2015).

$\alpha = 0.99$, $k = 2$ in the scalar $\varepsilon$-algorithm SEAW.
**First example:** They used as basic method, the **Cimmino method**.
They considered the **Lena** image of dimension $m = 128$, they blurred it (blur Matlab function from regtools of P.C.Hansen) obtaining a matrix of dimension $N = 16384$, and they added a white noise of $10^{-2}$.

In the following table we show the relative errors by comparing the Cimmino method, the **STEA2 (RM)**, and two possible restarts.

<table>
<thead>
<tr>
<th></th>
<th>100 iterations</th>
<th>$10 \times 10$ iterations</th>
<th>$20 \times 5$ iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cimmino</td>
<td>0.8074</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STEA2</td>
<td>0.1177</td>
<td>0.0913</td>
<td><strong>0.0841</strong></td>
</tr>
</tbody>
</table>

and for the images
Original image

Blurred, noisy image

Cimmino, 1000 iterations

STEA2, 20 × 5 iterations
Second example: They used as basic method, the Landweber method. They considered the satellite image (Restore Tools of J. Nagy et al) of dimension $m = 256$, obtaining a matrix of dimension $N = 65536$, with a Spatially Invariant Atmospheric Turbulence Blur and a noise level of $10^{-2}$. They compared STEA1 with different $y$ vectors (RM $20 \times 5$) and also with the GMRES method (after iteration 20 the method exhibits semiconvergence behaviour and the error grows quickly).
GMRES, 8 iterations

STEA1, 20 × 5 iterations
We think that our Matlab toolbox will be useful in trying to solve a very wide range of problems, in a pretty easy way:

- Acceleration of vector and matrix sequences
- Solution of linear and nonlinear vector, matrix and tensor equations
- Computation of matrix functions
- Solution of integral equations
- Image reconstruction and restoration
- ...............
We think that our Matlab toolbox will be useful in trying to solve a very wide range of problems, in a pretty easy way:

- Acceleration of vector and matrix sequences
- Solution of linear and nonlinear vector, matrix and tensor equations
- Computation of matrix functions
- Solution of integral equations
- Image reconstruction and restoration
- ...............

THANK YOU!