Acceleration, inexact Newton, and Nonlinear Krylov subspace methods

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

ICERM, Providence, RI
Aug. 31, 2015

Introduction

- Often need to accelerate a sequence of scalars or vectors $s_0, s_1, \cdots, s_n, ..$ that occur in some calculation
- For example: $s_{n+1} = g(s_n)$ [Fixed point iteration]
- or $s_n = \sum_{j=0}^{n} \alpha_j z^j$ [series]
- Various viewpoints adopted: Newton-type method, acceleration from Pade tables, Krylov projections, etc/
- Often the various communities are not familiar with each other's work..

Outline:

- Motivation
- Acceleration techniques
- Anderson acceleration
- Epsilon algorithm and related topics
- Nonlinear Krylov

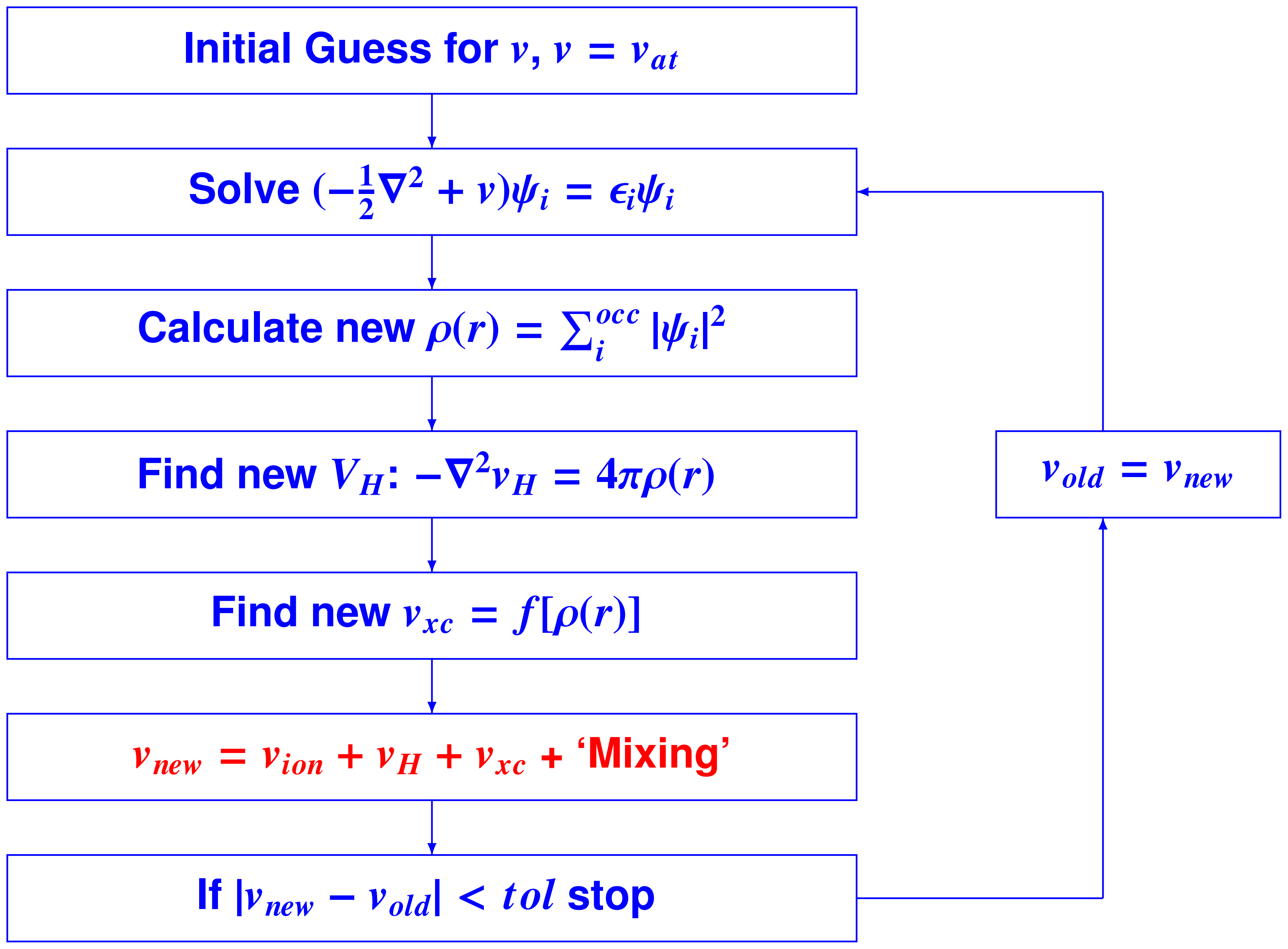
Kohn-Sham equations → a nonlinear eigenvalue problem

$$\left[-\frac{1}{2}\nabla^2 + (V_{ion} + V_H + V_{xc})\right]\Psi_i = E_i\Psi_i, i = 1, ..., n_o$$

$$\rho(r) = \sum_i^{n_o} |\Psi_i(r)|^2$$

$$\nabla^2 V_H = -4\pi\rho(r)$$

- Both V_{xc} and V_H, depend on ρ.
- Potentials & charge densities must be self-consistent.
- Broyden-type quasi-Newton technique used
- Most time-consuming part: diagonalization


Initial Guess for v, $v = v_{at}$
Solve $(-\frac{1}{2}\nabla^2 + v)\psi_i = \epsilon_i\psi_i$
Calculate new $\rho(r) = \sum_i^{occ} |\psi_i|^2$
Find new V_H: $-\nabla^2 v_H = 4\pi\rho(r)$
Find new $v_{xc} = f[\rho(r)]$
$v_{new} = v_{ion} + v_H + v_{xc}$ + 'Mixing'
If $|v_{new} - v_{old}| < tol$ stop
$v_{old} = v_{new}$

Matrix completion

Problem: A matrix $B \in \mathbb{R}^{m \times n}$ of small rank is partially known from its entries in some locations $(i, j) \in \Omega$. Task: fully recover B.

Example: Recommender systems. Matrix of ratings:

$$
\begin{array}{c|cccccccc}
 & \multicolumn{8}{c}{\text{Individuals}} \\
\hline
 & \star & \star & \star & ? & ? & \star & \star & ? \\
 & \star & ? & \star & \star & ? & \star & \star & \star \\
\text{Items} & ? & \star & \star & ? & \star & \star & \star & \star \\
 & \star & \star & \star & \star & ? & \star & \star & ? \\
 & \star & \star & \star & \star & \star & ? & \star & \star \\
 & \star & \star & ? & \star & \star & \star & \star & \star \\
\hline
\end{array}
$$

- Important problem in many other applications

- Let $\Omega = \{(i,j) | B_{ij}$ *is observed*$\}$ and $P_\Omega(X) \in \mathbb{R}^{m\times n}$ the projected matrix defined by

$$P_\Omega(X)_{ij} = \begin{cases} X_{ij} & \text{if}(i,j) \in \Omega \\ 0 & \text{otherwise.} \end{cases}$$

- Wish: *find matrix of smallest rank s.t.* $P_\Omega(X) = P_\Omega(B)$
- Hard problem

- Alternative: $$\min_{\{X \mid \text{rank}(X)=k\}} \|P_\Omega(X) - P_\Omega(B)\|_F$$
- Several algorithms developed.
- In some cases convergence is slow and acceleration has been advocated.
- Consider a form of subspace iteration for the SVD of B
- B is not known - but can use its latest approximation
- Start : truncated rank-k SVD approximation of $P_\Omega(B)$

ALGORITHM : 1 *Subspace Iteration for incomplete matrices.*

```
1. Initialize: $[U_0, S_0, V_0] = svd_k(P_\Omega(B)),\ X_0 = U_0 S_0 V_0^T$;
2. For $i = 0,1,2,...,$ Do:
3.     $X_{i+1} = X_i + t_i E_i$ where $E_i = P_\Omega(B - X_i)$
4.     $U_{i+1} = qf(X_{i+1} V_i)$;    $V_{i+1} = qf(X_{i+1}^T U_{i+1})$
5.     $S_{i+1} = U_{i+1}^T X_{i+1} V_{i+1}$;    $X_{i+1} := U_{i+1} S_{i+1} V_{i+1}^T$.
6. EndFor
```

From Line 4 easy to show that:

$$\boxed{V_{i+1} = \text{qf}(X_{i+1}^T X_{i+1} V_i)}$$

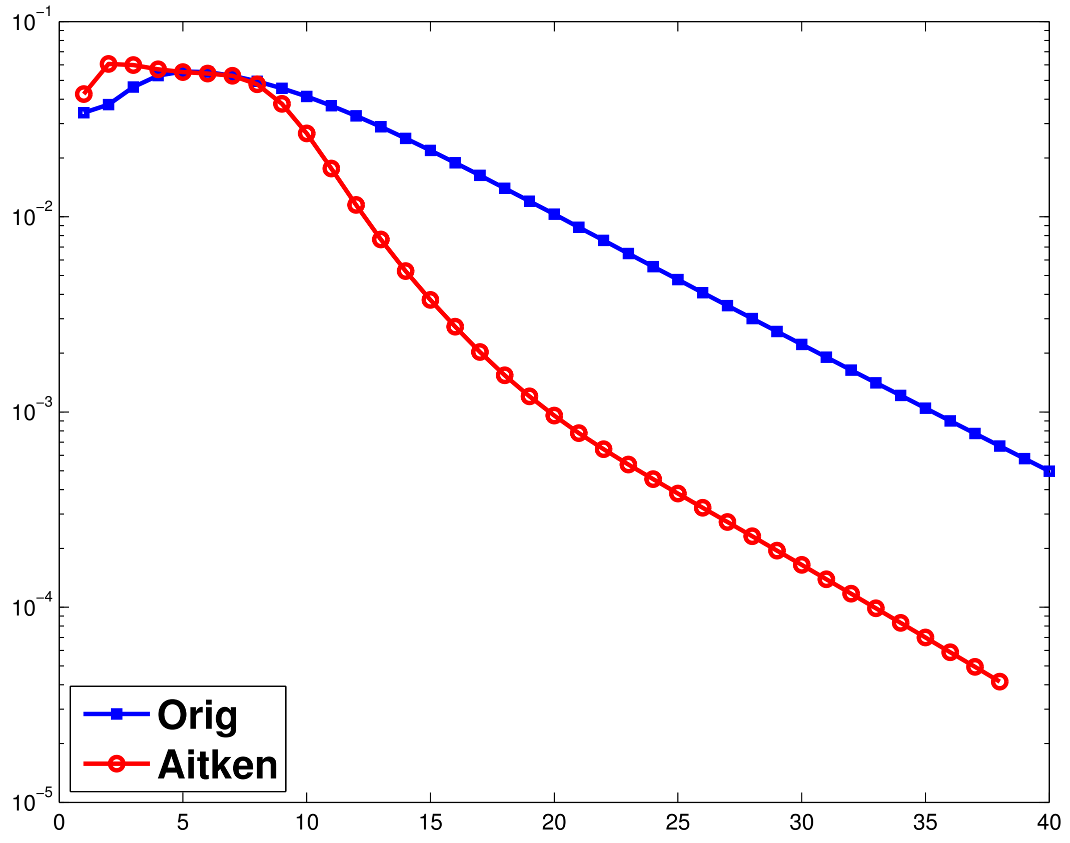
- Standard subspace iteration step applied to $X_{i+1}^T X_{i+1}$.

An example: $(m = 128, n = 16, p = 8)$

```
X = had(m,p)/sqrt(m);  % First p columns of Hadamard
                       % matrix (X'*X == I)
X = X*diag([1:p]/p);   % Resets singular values
u = [1:n]/(2*n+1);  v = [1:p];
B = X*cos(2*pi*v'*u);  % Mix X with a cosine-type transform
                       % B is now 128 × 16 of rank p = 8
```

- Create a 'mask' matrix C (sparse) – then
- incomplete matrix: `X = B .* C` [zero-out all those entries b_{ij} for which $C_{ij} = 0$]
- result: 293 entries (out of 2048) are lost (replaced by zeros).
- Goal: recover original B

- Let u_1 = dominant singular vector of B,
  $u_1^{(i)}$ = dominant singular vector of X_i,
- Plot error $||u_1 - u_1^{(i)}||$
- + $||error||$ resulting from Aitken acceleration



What is acceleration (or extrapolation)?

(Brezinski,'97) "Let (s_n) be a sequence of vectors (...). An extrapolation method is a method whose purpose is to acceleration the convergence of (s_n). It consists of transforming this sequence into a set of new sequences $\left(t_k^{(n)}\right)$ given by

$$\left(t_k^{(n)}\right) = a_0 s_n + a_1 s_{n+1} + \cdots + a_k s_{n+k}$$

where the a_i's can depend on k and n."

- We must have $a_0 + a_1 + \cdots a_k = 1$
- If s is the limit, we also would like to have

$$\boxed{\lim_{n\to\infty} \frac{\|t_k^{(n)} - s\|}{\|s_k^{(n)} - s\|} = 0}$$

A few historal landmarks

- ➤ Richardson's 'deferred approach to the limit' (h^2 extrapolation) – 1910. Used for discretized problems
- ➤ Aitken [1926] – initially to compute zeros of polynomials.
- ➤ Romberg [1955] – integration, ...
- ➤ Shanks [1949] generalizes Aitken's method
- ➤ Turning point: Wynn [1956] – found an elegant recursion to compute Shanks' transformation: the ϵ-algorithm
- ➤ Discovery ignited substantial following starting in the late late 1960s - early 1970s

- C. Brezinski, H. Sadok, K. Jbilou, M. Redivo Zaglia, Germain-Bonne, G. Walz, ...
- A. Sidi and co-workers ...
- Cabay-Jackson, Mesina, Kaniel-Stein, Mc-Leod, ...
- In physics: Different approaches - e.g., Anderson mixing, DIIS, ..., were developed - with a similar goal
- Viewpoint closer to quasi-Newton than to extrapolation
- Acceleration for linear systems : Chebyshev acceleration (old), but also Minimal Polynomial Extrapolation (MPE- Cabay-Jackson); Reduced Rank Extrapolation, many others
- A maze of connections and interesting (and often complex) results !

Example: Aitken δ^2 acceleration

Let s_{n-1}, s_n, s_{n+1} be given. And let $\lim_{n\to\infty} = s$

Assumption behind Aitken's process is that sequence satisfies

$$\boxed{s_{n+1} - s = \lambda(s_n - s) \quad \forall n}$$

- These sequences form the 'Kernel' of Aitken's process.
- λ, and s determined from s_n, s_{n+1}, s_{n+2} by writing:

$$\frac{s_{n+1}-s}{s_n-s} = \lambda, \quad \frac{s_{n+2}-s}{s_{n+1}-s} = \lambda \quad \to \lambda = \frac{s_{n+2}-s_{n+1}}{s_{n+1}-s_n} \quad \text{and:}$$

$$\boxed{s = \frac{s_n s_{n+2} - s_{n+1}^2}{s_{n+2} - 2s_{n+1} + s_n} = s_n - \frac{(\Delta s_n)^2}{\Delta^2 s_n}}$$

- Here Δ = forward difference operator ($\Delta x_i = x_{i+1} - x_i$).

Generalization: Shanks transform & the ϵ-algorithm

- Kernel for Aitken's process is of the form

$$a_1(s_n - s) + a_2(s_{n+1} - s) = 0 \ \forall n$$

- What if we generalize this to

$$a_1(s_n - s) + a_1(s_{n+1} - s) + \cdots + a_k(s_{n+k} - s) = 0 \ ?$$

- Consider $a_1, \cdots, a_{k+1}, s$ as unknowns.
- Write above for $n, n+1, \cdots, n+k$. Then add constraint:

$$a_1 + a_2 + \cdots + a_{k+1} = 1$$

$(k+2) \times (k+2)$ system.

$$\boxed{\begin{array}{lll} \sum_{j=1}^{k+1} a_j & & = 1 \\ \sum_{j=1}^{k+1} a_j s_{n+j+i-1} & -s & = 0, i = 0, .., k \end{array}}$$

- Get s using Cramer's rule
- Result: The Shanks (or 'Schmidt-Shanks') Transformation for scalar sequences

$$t_k^{(n)} = \frac{\begin{vmatrix} s_n & s_{n+1} & \cdots & s_{n+k} \\ \Delta s_n & \Delta s_{n+1} & \cdots & \Delta s_{n+k} \\ \vdots & \vdots & \vdots & \vdots \\ \Delta s_{n+k-1} & \Delta s_{n+k} & \cdots & \Delta s_{n+2k-1} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ \Delta s_n & \Delta s_{n+1} & \cdots & \Delta s_{n+k} \\ \vdots & \vdots & \vdots & \vdots \\ \Delta s_{n+k-1} & \Delta s_{n+k} & \cdots & \Delta s_{n+2k-1} \end{vmatrix}}$$

➤ Generalization of Aitken's δ^2 process ($k = 1$):

$$t_2^{(n)} = \frac{\begin{vmatrix} s_n & s_{n+1} \\ \Delta s_n & \Delta s_{n+1} \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ \Delta s_n & \Delta s_{n+1} \end{vmatrix}} = s_n - \Delta s_n (\Delta^2 s_n)^{-1} \Delta s_n$$

Q: How do we compute $t_k^{(n)}$?

➤ Determinants involve Hankel matrices → recurrences but complex and not numerically viable.

➤ P. Wynn [1956]: Simple recurrence relation to compute $t_k^{(n)}$ - known as the ϵ-algorithm

➤ From Ford, Smith & Sidi, SIAM review paper,'87:

“With a remarkable burst of insight Wynn[] discovered very soon after Shanks [] published his paper (...), that the required ratio of determinants could be evaluated recursively for increasing k and n without the use of determinants”

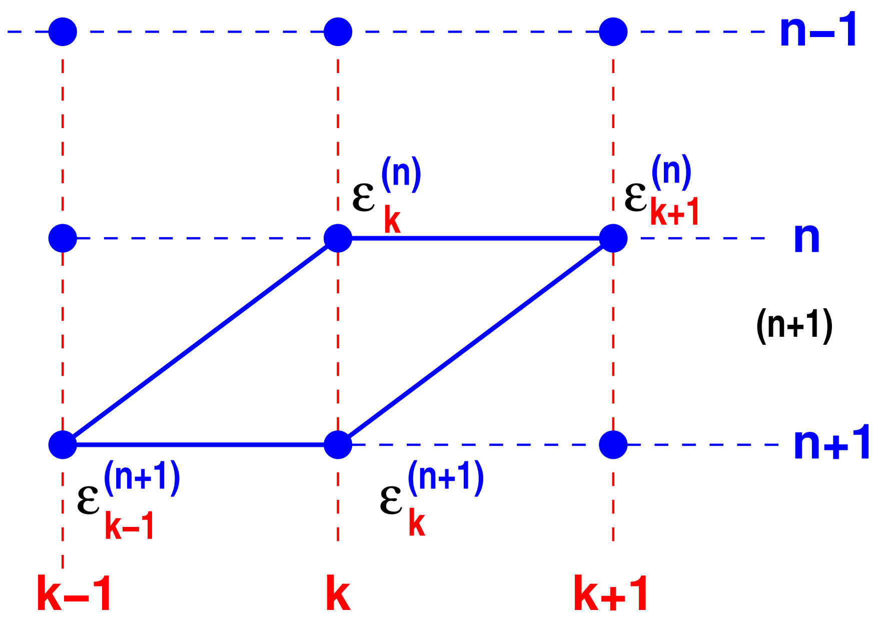
➤ From Brezinki's book [Pade-type approximation and general ortho. polynomials, '80, p. 162]:

“Wynn's proof of this equivalence between the ϵ algorithm and Shanks transformation was quite tedious; it involved very unusual properties of determinants. The same proof has now been achieved without any special knowledge of these properties but, of course (...) a proof is always easier when both the starting point and the end point are known”

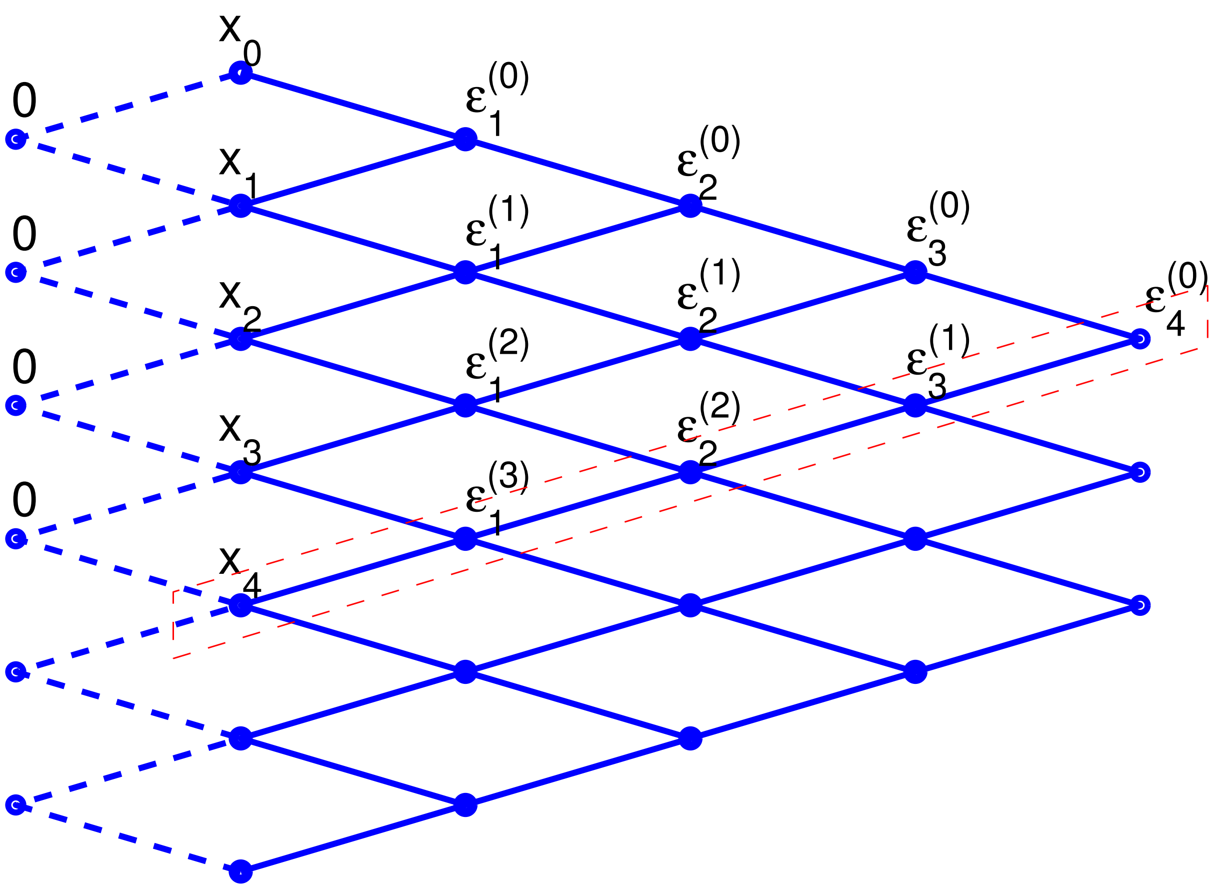
Start with $\epsilon_{-1}^{(0)} = 0$,
$\epsilon_{0}^{(n)} = s_n$ - then:

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

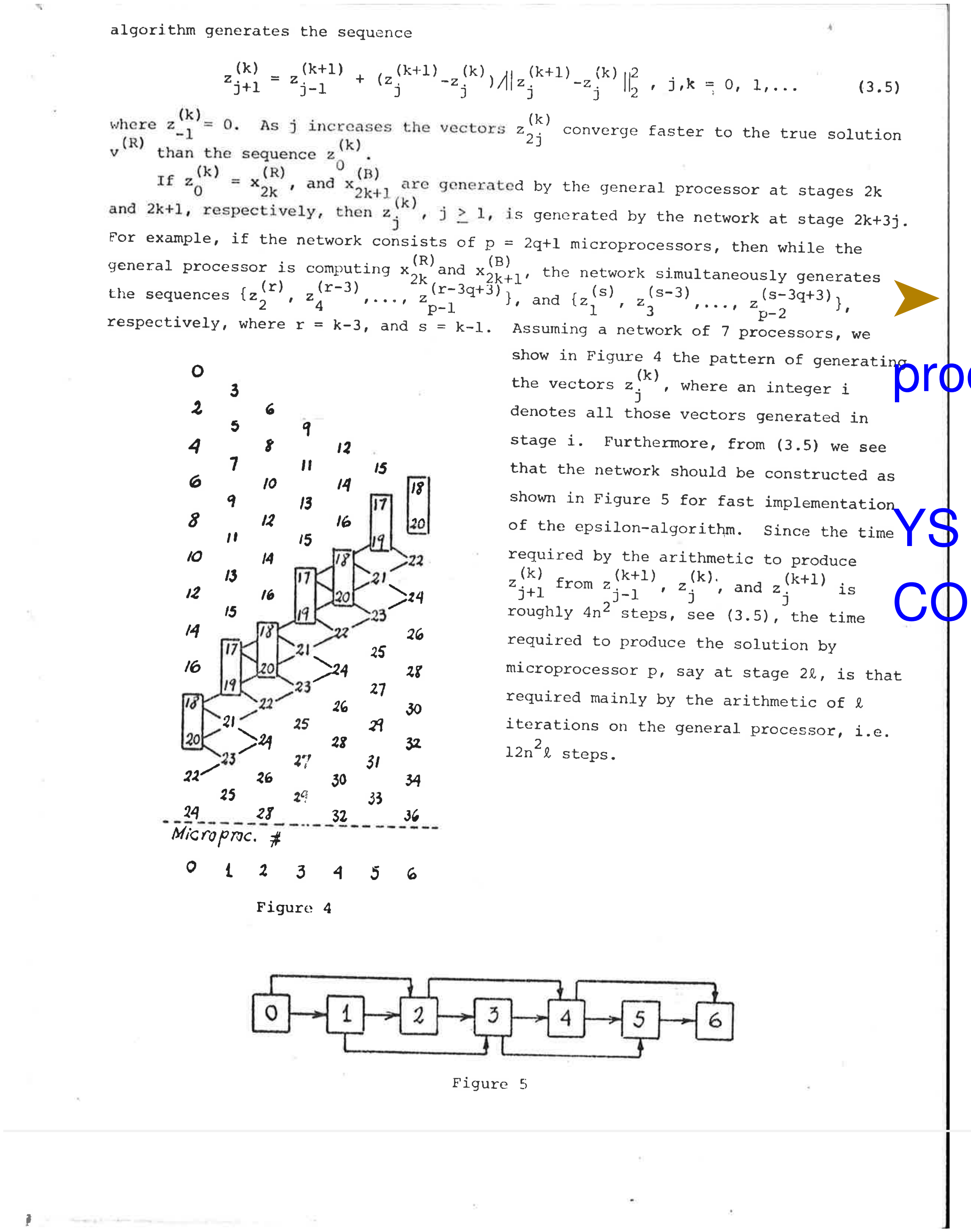
- $\epsilon_{2k} = e_k(s_n)$ = Shanks transform of s_n,
- $\epsilon_{2k+1} = 1/[e_k(s_n)]$ = auxilliary variables



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



- Practical implementation: Once x_n is available compute $\epsilon_1^{(n-1)}, \epsilon_2^{(n-2)}, \cdots, \epsilon_k^{(n-k)}$
- More stable variants exist [Brezinski, Redivo-Zaglia, '91]



algorithm generates the sequence

$$z_{j+1}^{(k)} = z_{j-1}^{(k+1)} + (z_j^{(k+1)} - z_j^{(k)})/\|z_j^{(k+1)} - z_j^{(k)}\|_2^2 , \quad j,k = 0, 1, \ldots \tag{3.5}$$

where $z_{-1}^{(k)} = 0$. As j increases the vectors $z_{2j}^{(k)}$ converge faster to the true solution $v^{(R)}$ than the sequence $z_0^{(k)}$.

If $z_0^{(k)} = x_{2k}^{(R)}$, and $x_{2k+1}^{(B)}$ are generated by the general processor at stages 2k and 2k+1, respectively, then $z_j^{(k)}$, $j \geq 1$, is generated by the network at stage 2k+3j. For example, if the network consists of p = 2q+1 microprocessors, then while the general processor is computing $x_{2k}^{(R)}$ and $x_{2k+1}^{(B)}$, the network simultaneously generates the sequences $\{z_2^{(r)}, z_4^{(r-3)}, \ldots, z_{p-1}^{(r-3q+3)}\}$, and $\{z_1^{(s)}, z_3^{(s-3)}, \ldots, z_{p-2}^{(s-3q+3)}\}$, respectively, where r = k-3, and s = k-1. Assuming a network of 7 processors, we show in Figure 4 the pattern of generating the vectors $z_j^{(k)}$, where an integer i denotes all those vectors generated in stage i. Furthermore, from (3.5) we see that the network should be constructed as shown in Figure 5 for fast implementation of the epsilon-algorithm. Since the time required by the arithmetic to produce $z_{j+1}^{(k)}$ from $z_{j-1}^{(k+1)}$, $z_j^{(k)}$, and $z_j^{(k+1)}$ is roughly $4n^2$ steps, see (3.5), the time required to produce the solution by microprocessor p, say at stage 2ℓ, is that required mainly by the arithmetic of ℓ iterations on the general processor, i.e. $12n^2\ell$ steps.

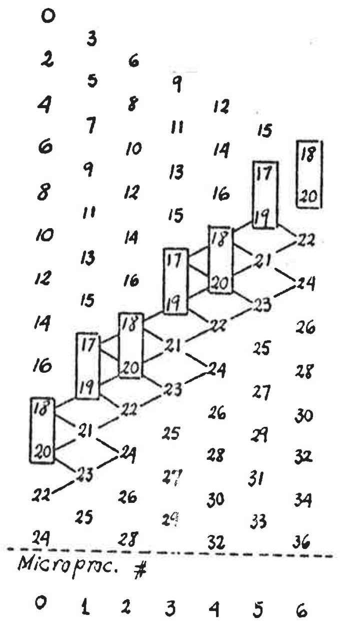


Figure 4

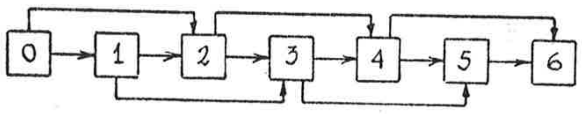


Figure 5

➤ Great for parallel processing too..

YS and A. Sameh, COMPAR'81

Vector ϵ-algorithms

- Easiest generalization define inverse of a vector:

$$v^{-1} \equiv \frac{v}{\|v\|^2}$$

- Another version [Brezinski]: Topological ϵ-algorithm (TEA)

$$\epsilon_{-1}^{(n)} = 0; \qquad \epsilon_0^{(n)} = x_n; \qquad y = \text{rand()}$$

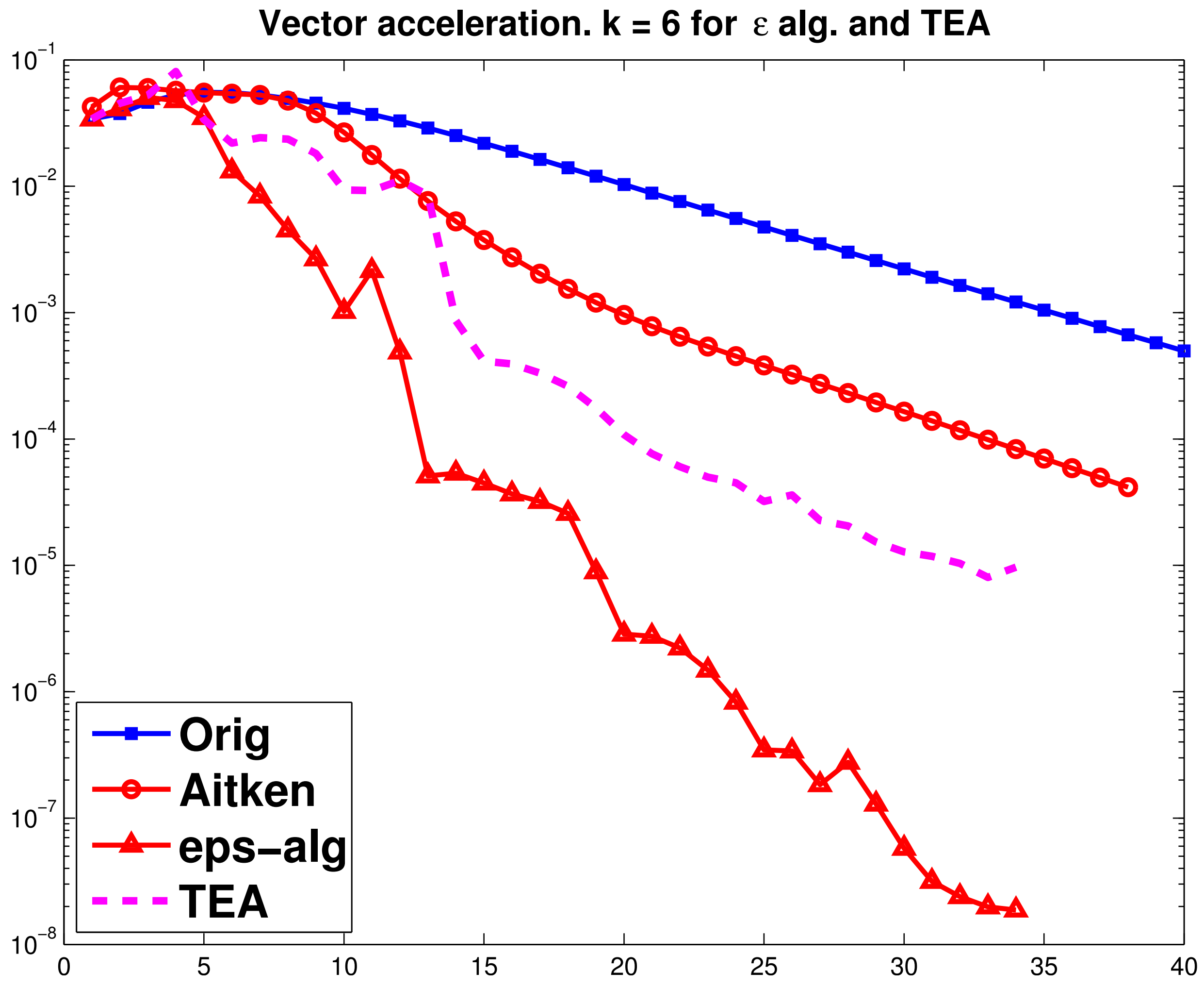
$$\epsilon_{2k+1}^{(n)} = \epsilon_{2k-1}^{(n+1)} + \frac{y}{(y, \epsilon_{2k}^{(n+1)} - \epsilon_{2k}^{(n)})}$$

$$\epsilon_{2k+2}^{(n)} = \epsilon_{2k}^{(n+1)} + \frac{\epsilon_{2k}^{(n+1)} - \epsilon_{2k}^{(n)}}{(\epsilon_{2k+1}^{(n+1)} - \epsilon_{2k+1}^{(n)}, \epsilon_{2k}^{(n+1)} - \epsilon_{2k}^{(n)})}$$

- Based on the definition: $v^{-1} \equiv y/(v, y)$.

- Corresponds to

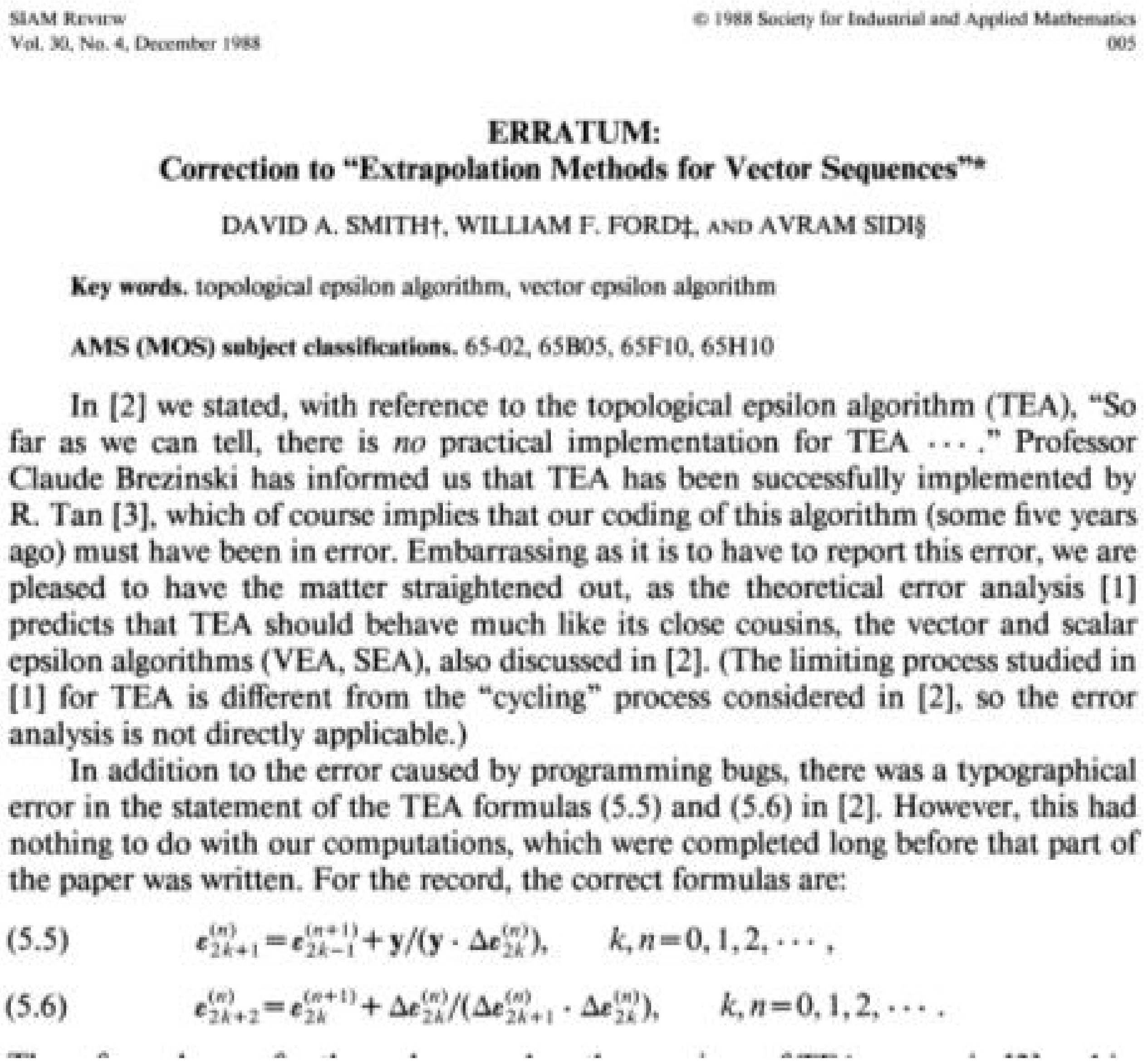
$$e_k^{(n)} = \frac{\begin{vmatrix} x_n & x_{n+1} & \cdots & x_{n+k} \\ (y, \Delta x_n) & (y, \Delta x_{n+1}) & \cdots & (y, \Delta x_{n+k}) \\ \vdots & \vdots & \vdots & \vdots \\ (y, \Delta x_{n+k-1}) & (y, \Delta x_{n+k}) & \cdots & (y, \Delta x_{n+2k-1}) \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ (y, \Delta x_n) & (y, \Delta x_{n+1}) & \cdots & (y, \Delta x_{n+k}) \\ \vdots & \vdots & \vdots & \vdots \\ (y, \Delta x_{n+k-1}) & (y, \Delta x_{n+k}) & \cdots & (y, \Delta x_{n+2k-1}) \end{vmatrix}}$$


Vector acceleration. k = 6 for ε alg. and TEA
10^−1
10^−2
10^−3
10^−4
10^−5
10^−6
10^−7
10^−8
0
5
10
15
20
25
30
35
40
Orig
Aitken
eps−alg
TEA

A few references

1. C. Brezinski, *Padé Type Approximation and General Orthogonal Polynomials*, Birkhäuser-Verlag, Basel-Boston-Stuttgart, 1980.
2. ——, *Projection Methods for Systems of Equations*, North-Holland, Amsterdam, 1997.
3. C. Brezinski and M. Redivo Zaglia, *Extrapolation Methods: Theory and Practice*, North-Holland, Amsterdam, 1991.
4. A. Sidi, *Extrapolation vs. projection methods for linear systems of equations*, J. of Comput. Appl. Math., 22 (1988), pp. 71–88.
5. D. A. Smith, W. F. Ford, and A. Sidi, *Extrapolation methods for vector sequences*, SIAM review, 29 (1987), pp. 199–233.

➤ Ahh Those nasty bugs



SIAM Review
Vol. 30, No. 4, December 1988

ERRATUM:
Correction to "Extrapolation Methods for Vector Sequences"*

DAVID A. SMITH†, WILLIAM F. FORD‡, AND AVRAM SIDI§

Key words. topological epsilon algorithm, vector epsilon algorithm

AMS (MOS) subject classifications. 65-02, 65B05, 65F10, 65H10

In [2] we stated, with reference to the topological epsilon algorithm (TEA), "So far as we can tell, there is *no* practical implementation for TEA $\cdots$." Professor Claude Brezinski has informed us that TEA has been successfully implemented by R. Tan [3], which of course implies that our coding of this algorithm (some five years ago) must have been in error. Embarrassing as it is to have to report this error, we are pleased to have the matter straightened out, as the theoretical error analysis [1] predicts that TEA should behave much like its close cousins, the vector and scalar epsilon algorithms (VEA, SEA), also discussed in [2]. (The limiting process studied in [1] for TEA is different from the "cycling" process considered in [2], so the error analysis is not directly applicable.)

In addition to the error caused by programming bugs, there was a typographical error in the statement of the TEA formulas (5.5) and (5.6) in [2]. However, this had nothing to do with our computations, which were completed long before that part of the paper was written. For the record, the correct formulas are:

(5.5) $$\varepsilon^{(n)}_{2k+1} = \varepsilon^{(n+1)}_{2k-1} + \mathbf{y}/(\mathbf{y}\cdot\Delta\varepsilon^{(n)}_{2k}), \qquad k, n = 0, 1, 2, \cdots,$$

(5.6) $$\varepsilon^{(n)}_{2k+2} = \varepsilon^{(n+1)}_{2k} + \Delta\varepsilon^{(n)}_{2k}/(\Delta\varepsilon^{(n)}_{2k+1}\cdot\Delta\varepsilon^{(n)}_{2k}), \qquad k, n = 0, 1, 2, \cdots.$$

The inexact Newton framework

To solve $F(u) = 0$ $(F : \mathbb{R}^N \rightarrow \mathbb{R}^N)$ we can use Newton iteration:

```
Set $u_0$ = an initial guess.
For $n = 0, 1, 2, \cdots$ until convergence do:
        Solve: $J(u_n)\delta_n = -F(u_n)$      (*)
        Set: $u_{n+1} = u_n + \delta_n$
```

where $J(u_n) = F'(u_n)$ = system Jacobian.

Inexact Newton methods: solve system (*) approximately.

Quasi-Newton methods: solve system (*) in which Jacobian is replaced by an estimate obtained from previous iterates.

Newton-Krylov methods: solve system (*) by a Krylov subspace method

Note: In Krylov-Newton, Jacobian of F not needed explicitly.

- To compute Jv use finite difference approximation:

$$\frac{\partial F}{\partial u} v \approx \frac{F(u+\epsilon v)-F(u)}{\epsilon}$$

- Newton-Krylov can be viewed as a special case of quasi-Newton approaches
- Can use this framework to accelerate sequences of the form

$$\boxed{u_{n+1} = M(u_n)}$$

.. by solving $F(x) = 0$ where $F(x) = M(x) - x$

Important difference with extrapolation techniques:

we now need to compute $F(u_n + \epsilon v)$ for arbitrary v,

- ... instead of using only the u_j's and f_j's generated ($f_j = f(u_j)$).

Anderson Acceleration

- Given: $x_j \qquad j = n-k, \cdots, n$

  $f_j = f(x_j) \qquad j = n-k, \cdots, n$
- Let: $\Delta x_i = x_{i+1} - x_i$

  $\Delta f_i = f_{i+1} - f_i,$

  $\mathcal{X}_n = [\Delta x_{n-k} \ \cdots \ \Delta x_{n-1}],$

  $\mathcal{F}_n = [\Delta f_{n-k} \ \cdots \ \Delta f_{n-1}],$

Anderson mixing: seek an 'accelerated' sequence in the form

$$\bar{x}_n = x_n - \sum_{i=n-k}^{n-1} \gamma_i^{(n)} \Delta x_i = x_n - \mathcal{X}_n \gamma_n, \qquad \text{with } \gamma_n = \begin{pmatrix} \gamma_{n-k}^{(n)} \\ \vdots \\ \gamma_{n-1}^{(n)} \end{pmatrix}$$

Notation issues: n is index of most recent iterate.

Read as: $x_{new} = x_{last} -$ Lin. Comb. previous Δx_i's

Anderson Acceleration

- γ_n obtained by writing:

$$\bar{f}_n = f_n - \sum_{i=n-k}^{n-1} \gamma_i^{(n)} \Delta f_i = f_n - \mathcal{F}_n \gamma_n,$$

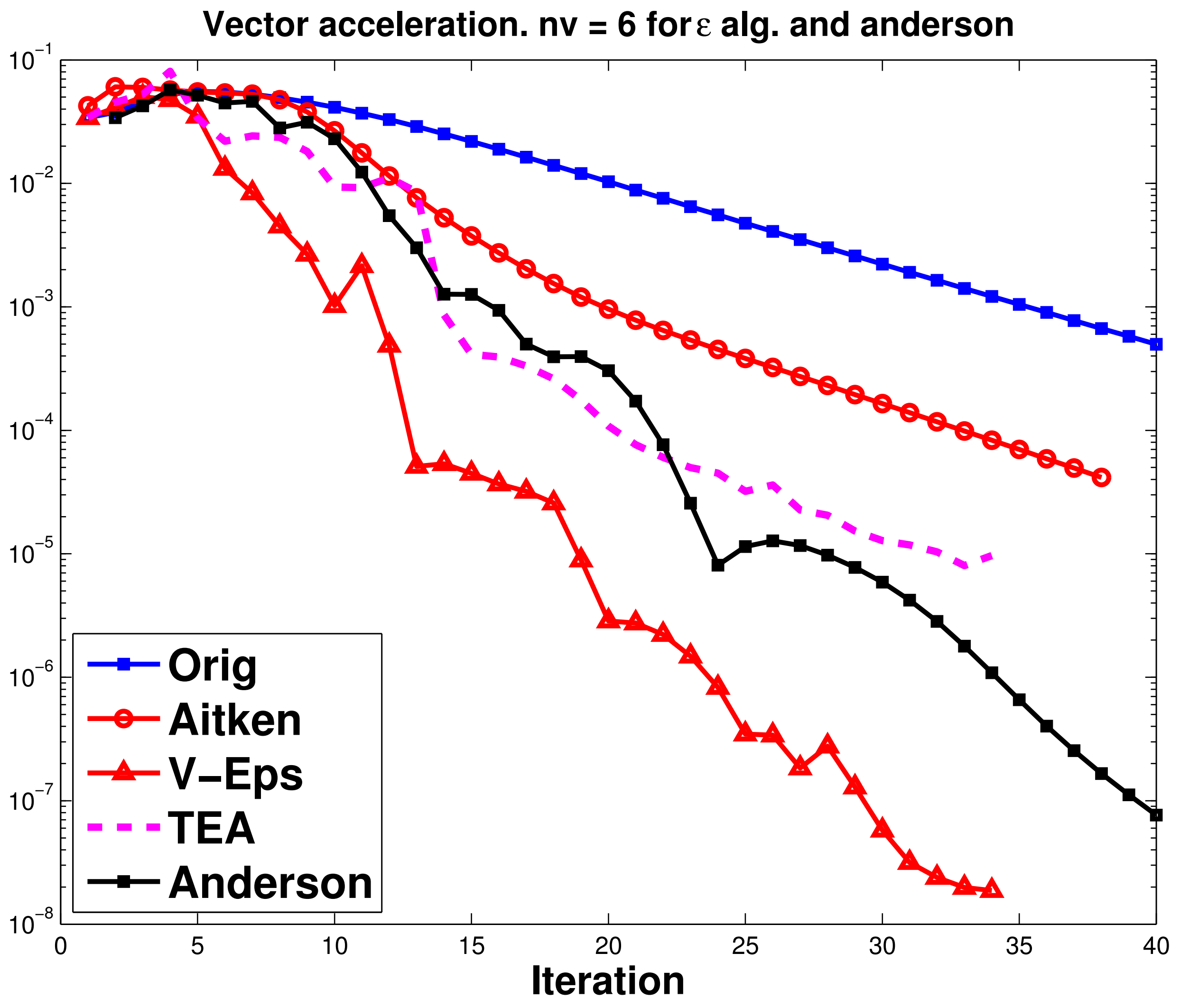
- So compute γ_n as

$$\gamma_n = \text{argmin}_\gamma \|f_n - \mathcal{F}_n \gamma\|_2$$

Note: Can formulate problem in the standard 'acceleration' form

$$\bar{x}_n = \sum_{i=n-k}^{n} \mu_i^{(n)} x_i \quad \text{with} \quad \sum \mu_i^{(n)} = 1$$

- Mathematically equivalent


Vector acceleration. nv = 6 for ε alg. and anderson
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
0
5
10
15
20
25
30
35
40
Iteration
Orig
Aitken
V−Eps
TEA
Anderson

Relation with other methods

➤ In “generalized Broyden methods” [Louis & Vanderbilt’84, Eyert’96] approximate Jacobian G_n satisfies m secant conditions:

$$\boxed{G_n \Delta f_i = \Delta x_i \text{ for } i = n-k, \ldots, n-1.}$$

➤ Matrix form: $\boxed{G_n \mathcal{F}_n = \mathcal{X}_n}$

➤ No-change condition:

$$(G_n - G_{n-k})q = 0 \quad \forall q \in \text{Span}\{\Delta f_{n-k}, \ldots, \Delta f_{n-1}\}^\perp$$

➤ After calculations we get a rank-k update formula:

$$G_n = G_{n-k} + (\mathcal{X}_n - G_{n-k}\mathcal{F}_n)(\mathcal{F}_n^T \mathcal{F}_n)^{-1}\mathcal{F}_n^T,$$

... and an update of the form:

$$\boxed{x_{n+1} = x_n - G_{n-k} f_n - (\mathcal{X}_n - G_{n-k}\mathcal{F}_n)\gamma_n; \quad \gamma_n = \mathcal{F}_n^{\dagger} f_n}$$

➤ Setting $G_{n-k} = -\beta I$ yields exactly Anderson's original method [which includes a parameter β]

➤ Result shown by Eyert (1996) [See also H-r Fang and YS]

➤ With $\beta = 0$ update is simply: $x_{n+1} = x_n - \mathcal{X}_n \mathcal{F}_n^{\dagger} f_n$

➤ Walker and Ni'11: equivalence with GMRES in linear case.

Q: Any relations to any one of the extrapolation techniques in nonlinear case?

Extrapolation by projection: MPE, RRE, MMPE

[See: *Jbilou and Sadok, Numer. Math. '95*]

- Given $x_n, \Delta x_n, \Delta^2 x_n,$

for $n = 0, 1, 2, \cdots$,

- Accelerated sequences: $\nearrow$

$$\boxed{t_n^{(k)} = \sum_{j=0}^{k} \alpha_j x_{n+j}}$$

- Coefficients α_j satisfy: $\searrow$

$$\boxed{\begin{array}{c} \sum_{j=0}^{k} \alpha_j = 1 \\ \sum_{j=0}^{k} \eta_{ij}\alpha_j = 0 \end{array}}$$

with $\eta_{ij} = (y_i, \Delta x_{n+j})$ and:

- $y_i = \Delta x_{n+i}$ for MPE
- $y_i = \Delta^2 x_{n+i}$ for RRE
- $y_i =$ vector$_i$ for MMPE
- For TEA $\eta_{ij} = (y, \Delta x_{n+i+j})$ where y = some vector

- Let us exclude TEA. Cramer's rule gives:

$$t_k^{(n)} = \frac{\begin{vmatrix} x_n & x_{n+1} & \cdots & x_{n+k} \\ (y_0, \Delta x_n) & (y_0, \Delta x_{n+1}) & \cdots & (y_0, \Delta x_{n+k}) \\ \vdots & \vdots & \vdots & \vdots \\ (y_{k-1}, \Delta x_n) & (y_{k-1}, \Delta x_{n+1}) & \cdots & (y_{k-1}, \Delta x_{n+k}) \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ (y_0, \Delta x_n) & (y_0, \Delta x_{n+1}) & \cdots & (y_0, \Delta x_{n+k}) \\ \vdots & \vdots & \vdots & \vdots \\ (y_{k-1}, \Delta x_n) & (y_{k-1}, \Delta x_{n+1}) & \cdots & (y_{k-1}, \Delta x_{n+k}) \end{vmatrix}}$$

- Let: $Y_k = [y_0, \cdots, y_{k-1}]$, $X_n = [x_n, \cdots x_{n+k-1}]$. Then

$$t_k^{(n)} = x_n - \Delta X_n (Y_k^T \Delta^2 X_n)^{-1} Y_k^T \Delta x_n$$

RRE: Let $n = 0$, $k = 0, 1, 2, ...$ and $U \equiv \Delta X_0; \; V \equiv \Delta^2 X_0$. Then:

$$t_k^{(0)} = x_0 - \Delta X_0 (\Delta^2 X_0)^\dagger \Delta x_0 = x_0 - UV^\dagger \Delta x_0$$

Claim: $t_k^{(0)} = x_j - UV^\dagger \Delta x_j$ *for any* $0 \le j \le k$

Proof for $j = 3$: Note that

$$\begin{aligned}\Delta x_0 &= \Delta x_3 - (\Delta x_3 - \Delta x_2) - (\Delta x_2 - \Delta x_1) - (\Delta x_1 - \Delta x_0)\\ &= \Delta x_3 - \Delta^2 x_2 - \Delta^2 x_1 - \Delta^2 x_0\end{aligned}$$

and that $(UV^\dagger)\Delta^2 x_i = \Delta x_i$ since

$\Delta X_0 [(\Delta^2 X_0^T \Delta^2 X_0)^{-1} \Delta^2 X_0^T] \Delta^2 X_0 = \Delta X_0.$ In the end:

$$\begin{aligned}x_0 - UV^\dagger \Delta x_0 &= x_0 - UV^\dagger [\Delta x_3 - \Delta^2 x_2 - \Delta^2 x_1 - \Delta^2 x_0]\\ &= x_0 - UV^\dagger \Delta x_3 + (x_3 - x_2) + (x_2 - x_1) + (x_1 - x_0)\\ &= x_3 - UV^\dagger \Delta x_3 \qquad \square\end{aligned}$$

Reduced Rank Extrapolation (RRE) – rewritten

- Mesina'77, Eddy'79, Kaniel and Stein'74,
- Developed for linear systems – Used for nonlinear

0. Given $x_0, x_1, x_2, \cdots,$
1. Compute $u_j = \Delta x_j$, $v_j = \Delta u_j$, $j = 0, \cdots, k-1$.
2. Let: $U \equiv U_k = [u_0, u_1, \cdots, u_{k-1}]$
   $V \equiv V_k = [v_0, v_1, \cdots, v_{k-1}]$
3. Compute $\bar{x}_k = x_0 - UV^\dagger u_0$

Recall: Can replace line 3 by $\bar{x}_k = x_j - UV^\dagger u_j$ for $0 \le j \le k \rightarrow$

$$\bar{x}_k = x_k - UV^\dagger \Delta x_k$$

- For fixed point mappings $x_{j+1} = M(x_j)$ then $\Delta x_j = M(x_j) - x_j$.
- Let $F(x) = M(x) - x$ Then

$$U = [\Delta x_0, \Delta x_1, \cdots, \Delta x_{k-1}],$$
$$V = [\Delta f_0, \Delta f_1, \cdots, \Delta f_{k-1}],$$

- compare results with those of Anderson: $\bar{x}_k = x_k - \mathcal{X}_k \mathcal{F}_k^{\dagger} f_k$
- in both cases:

$$\bar{x}_k = x_k - UV^{\dagger} f_k$$

Conclusion: *RRE is mathematically equivalent to Anderson's acceleration.* (even in nonlinear case).

- ➤ Several other connections exist
- ➤ In linear case:
- ➤ Brezinski [1980] showed that TEA is equivalent to Lanczos algorithm
- ➤ A. Sidi ['88] showed several equivalences between Krylov subspace methods and 'polynomial acceleration' methods [MMPE, RRE, TEA]. Also shown by Beuneu'84 [unpublished report]. See also NumMath paper by Jbilou & Sadok'95.
  - RRE is mathematically equivalent to GMRES.
  - MPE is equivalent to Arnoldi (FOM)
  - TEA is equivalent to Lanczos (biCG)

Conclusion

- A huge variety of acceleration techniques developed over the years
- Many are mathematically equivalent to other methods in linear case
- Not clear which work best in nonlinear case in specific situations
- Quasi-Newton [and “generalized Broyden”] is a fairly comprehensive viewpoint.