



ARES: a crude but efficient approach to adaptive time integration for stiff Differential-Algebraic Equations

N. Leterrier

► To cite this version:

N. Leterrier. ARES: a crude but efficient approach to adaptive time integration for stiff Differential-Algebraic Equations. Recent Trends in Applied Mathematics RTM'19, Apr 2019, Oxford, United Kingdom. cea-02614119

HAL Id: cea-02614119

<https://cea.hal.science/cea-02614119>

Submitted on 20 May 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ARES: a crude but efficient approach to adaptive time integration for stiff Differential-Algebraic Equations



N. Leterrier

CEA Saclay

Outline

- 1 General context
- 2 **Adaptive Relaxed Euler Scheme**
- 3 Why it works in theory
- 4 Numerical results
- 5 A good rule of thumb
- 6 Conclusions

In many industrial or natural configurations, involving heat and reactive transport we face :

Stiff Nonlinear Differential-Algebraic Equations

→ *Chemical processes, Nuclear PWR clogging, Radioactive waste recycling or disposal, Hydrogeology, CO₂ underground storage...*

Stiffness of a solution curve may result from chemical reactions depending on temperature, catalytic processes or complex fluxes etc...

When looking for the **steady state** of a system, through a transient simulation with stiff phenomena, **adaptive time integration** then becomes a necessity, to adapt the timestep to local stiffness.

2 broad categories of time integrators are often used :

- BDF (backwards differentiation formula) and MEBDF (modified extended backwards differentiation formula) approaches (Gear, DASSL or MEBDFDAE algorithms)
- Runge-Kutta approaches

High order time discretization $\Rightarrow$ costly additional operations at each iteration of the nonlinear solver

What if you **only care for a precise evaluation of the steady state**, however precise the transient way to get there may be ?

$\longrightarrow$ **ARES** may be of service.

*You are the god(dess) of war aiming at a swift victory
over your nonlinear foe !*

→ Let $F : (\mathbb{R}, \mathbb{R}^p, \mathbb{R}^p) \rightarrow \mathbb{R}$ be a continuously differentiable nonlinear function. A typical DAE solution $y : \mathbb{R} \rightarrow \mathbb{R}^p$ writes :

$$F\left(t, y, \frac{dy}{dt}\right) = 0$$

$$\forall t, \forall i \in \llbracket 1, p \rrbracket, l_i \leq y_i(t) \leq u_i$$

→ Euler implicit time integration scheme means that we solve at every t_n :

$$F\left(t_n, y_n, \frac{y_n - y_{n-1}}{h_n}\right) = 0$$

→ Implicit Euler time schemes with an adaptive timestep differ mainly by the strategy governing the choice of the next timestep $h_n = t_n - t_{n-1}$.

→ As is done very often in adaptive time schemes, we use a predictor to propose a prediction of y_n , noted as $y_n^*(o)$, based upon a Taylor evaluation of order $o = 1$ or $o = 2$

$$\begin{cases} y_n^*(1) &= y_{n-1} + d_{n-1}h_n \text{ with } d_{n-1} = \frac{y_{n-1} - y_{n-2}}{h_{n-1}} \\ y_n^*(2) &= y_n^*(1) + \frac{1}{2}d_{n-1}^2h_n^2 \text{ with } d_{n-1}^2 = \frac{d_{n-1} - d_{n-2}}{h_{n-1}} \end{cases}$$

→ $y_n^*(o)$ is used as the initial guess for a classical Newton method.

→ We monitor the behaviour of the solution using the number of iterations used by the Newton method (noted henceforth I_n) to converge towards the solution. This value is a direct product of the Newton solver and **needs no further calculation.**

A very basic and simple idea :

→ The Newton solver's number of iterations is viewed here as an indicator of the gap between y_n and the predicted value $y_n^*(o)$.

→ This gap between y_n and $y_n^*(o)$ is informally related to the stiffness of the solution.

→ We allow the timestep to increase when this number is low, because the solution should be smooth enough between t_{n-1} and t_n . Conversely, we decrease the timestep when this number is high.

→ Since we aim only for the steady state : $F(t, y, 0) = 0$, high order time discretization of $\frac{dy}{dt}$ is nullified in the end and order 1 is sufficient.

- l_n : number of iterations needed to compute y_n
- l_u : a user-defined "ideal" number of iterations
- $\gamma_a > 1$: acceleration rate
- $0 < \gamma_d < 1$: deceleration rate

At first, we take $h_n = h_{n-1}$ and then we apply the following choices, naming h_n^* the previous value of h_n :

- Convergence fails $\longrightarrow h_n = \gamma_d h_n^*$
- Convergence succeeds in l_n iterations :
 - ▶ $l_n = l_u \longrightarrow$ next timestep with $h_{n+1} = h_n^*$
 - ▶ $l_n > l_u \longrightarrow$ treated as failure
 - ▶ $l_n < l_u \longrightarrow$ next timestep with $h_{n+1} = \max(\gamma_a h_n^*, h_{\max})$

Delayed version : $l_n > l_u$ is not treated as a failure, but the timestep is nevertheless reduced : $h_{n+1} = \gamma_d h_n^$.*

ARES fully solves at each timestep the equation $F(t_n, y_n, \frac{y_n - y_{n-1}}{h_n}) = 0$, through the Newton method, at any requested precision.

Stationary time : $y_n - y_{n-1} = 0 \implies F(t_n, y_n, 0) = 0$

$\longrightarrow y_n$ solves the exact equation of the steady state equation (provided of course that this solution remains within the boundaries).

ARES can also be used for a transient calculation, with $h < h_{max}$ to ensure a minimal precision, since the order of the time scheme remains 1.

Let $G : \mathbb{R}^p \longrightarrow \mathbb{R}^p$ be the following function :

$$G(y) = F(t_n, y, \frac{y - y_{n-1}}{h_n}) \quad (1)$$

At every timestep t_n , we search for y_n as the solution of $G(y) = 0$ through the Newton method. Let $J(y)$ be the jacobian of G for y . One Newton iteration (step k) verifies :

$$J(y^k)(y^{k+1} - y^k) = -G(y^k) \quad (2)$$

If we decompose $y = [y_A, \dots, y_p]$ and $G(y) = [g_1(y), \dots, g_p(y)]$ accordingly for a p -sized system, we have :

$$\sum_{j=1}^p (y_j^{k+1} - y_j^k) \frac{\partial g_i}{\partial y_j}(y_A^k, \dots, y_p^k) = -g_i(y_A^k, \dots, y_p^k) \quad \forall i \in [1, p] \quad (3)$$

Let $z : \mathbb{R} \rightarrow \mathbb{R}^p$ be solution of the following differential equation :

$$\frac{d}{dt}[g_i(z(t))] = -g_i(z(t)) \quad (4)$$

$\rightarrow \forall i \in [1, p]$ (3) can be seen as an **implicit Euler time discretization** of (4), with a constant timestep=1.

The solution $z(t)$ ("Newton flow") verifies :

$$g_i(z(t)) = g_i(z(0)) \exp^{-t} \quad (5)$$

If the Newton iterations run by (3) converge towards the solution y_n , there is by construction a discret solution of (4) $\forall i \in [1, p]$.

The values of $z(t)$ form then a trajectory towards y_n , with a constant timestep.

According to (5), the trajectory depends only on the values of $g_i(z(0))$ or **the first iterate of the Newton method**.

→ Direct correspondence between the number of Newton iterations l_n and the overall closeness of $G(y_n^*(o))$ to zero.

→ Preserving a low l_n ensures then a relative smoothness of the solution between t_{n-1} and t_n .

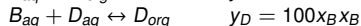
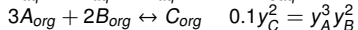
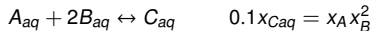
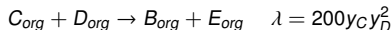
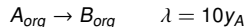
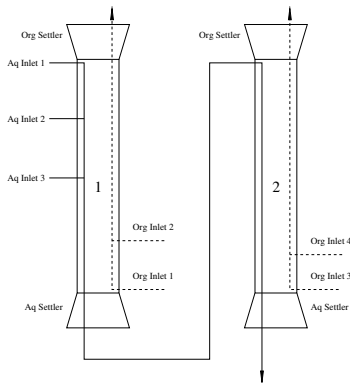
SIDES is a module of the open source platform TRUST, dedicated to thermohydraulics in the context of nuclear plant simulations :

- C++ platform providing many numerical tools, such as PETSc linear solvers, pre- and postprocessing tools etc...
- Basis for TrioCFD, Genepi+, Cathare...
- SIDES aggregates objects associated with numerical methods : being given a nonlinear function of multiple variables, its jacobian and a set of bounds, it may be used to solve any $F\left(t, y, \frac{dy}{dt}\right) = 0$

→ ARES was used in SIDES for an industrial code dedicated to Liquid-Liquid-Extraction :

- reactive and heat transport between an organical and an aqueous phase countercurrently,
- both intra- and interphasic chemical reactions,
- chemical reactions under kinetics or at equilibrium,
- retroaction of chemistry on temperature.

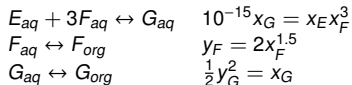
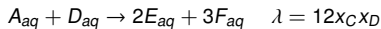
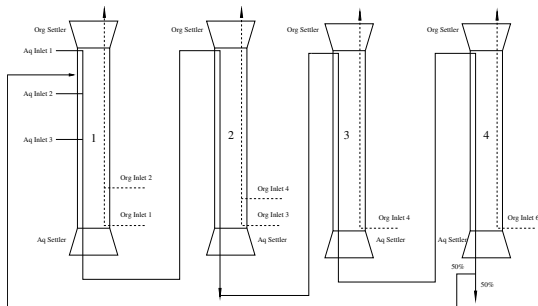
2 Pulsed Columns, 2 chemical phases with 11 fictitious species



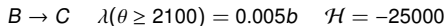
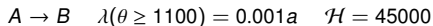
Density laws : ρ_{aq}, ρ_{org}

Variable fluxes with phase swelling

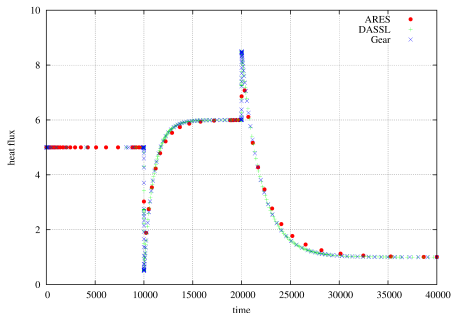
4 Pulsed Columns, 5 more fictitious species, 1 more kinetics and 3 more equilibria



Analytical case study : 2 fast reactions, with a threshold temperature.



Heating device : $\theta(t) = 100 + 0.1t$. We monitor the needed heat flux Φ :



→ Steady plateaux $\Phi = 5$, $\Phi = 6$ and $\Phi = 1$ accurately given by ARES, with much less simulation times.

→ As expected : Gear or DASSL more accurate on transient regimes.

method	Gear	DASSL	ARES	ARES delayed
case study 1	114	172	78	81
case study 2	294	492	295	189
case study 3	3.6	6	0.8	0.8

How to choose an ideal number of iterations l_u ?

A DAE system contains both algebraic and differential equations :

- For differential equations, decreasing the timestep hopefully means that the overall function giving y^{k+1} in relation to y^k may become **contractive**.
- However, the presence of purely algebraic equations, independent from time derivatives, invalidates or at least challenges this strategy.

Let us consider a DAE system containing only one algebraic equation :

$$\begin{bmatrix} y_A - y_A^{n-1} - h_n f_1(y) \\ \vdots \\ y_{p-1} - y_{p-1}^{n-1} - h_n f_{p-1}(y) \\ f_p(y) \end{bmatrix} = 0 \quad (6)$$

If we decompose the jacobian $J = J_1 + h_n J_2$, where J_1 gathers all terms independant of h_n , the Newton step writes :

$$\begin{bmatrix} y_A^k - y_A^{n-1} - h_n f_1(y^k) \\ \dots \\ y_{p-1}^k - y_{p-1}^{n-1} - h_n f_{p-1}(y^k) \\ f_p(y^k) \end{bmatrix} = \begin{bmatrix} (1 + \frac{\partial f_p}{\partial y_A}(y^k))(y_A^k - y_A^{k+1}) \\ \dots \\ (1 + \frac{\partial f_p}{\partial y_{p-1}}(y^k))(y_{p-1}^k - y_{p-1}^{k+1}) \\ \sum_{j=1}^p \frac{\partial f_p}{\partial y_j}(y^k)(y_j^k - y_j^{k+1}) \end{bmatrix} + h_n J_2[y^k - y^{k+1}] \quad (7)$$

This simplifies as :

$$\begin{bmatrix} -y_A^{n-1} - h_n f_1(y^k) \\ \dots \\ -y_{p-1}^{n-1} - h_n f_{p-1}(y^k) \\ f_p(y^k) \end{bmatrix} = -y^{k+1} + \begin{bmatrix} \frac{\partial f_p}{\partial y_A}(y^k)(y_A^k - y_A^{k+1}) \\ \dots \\ \frac{\partial f_p}{\partial y_{p-1}}(y^k)(y_{p-1}^k - y_{p-1}^{k+1}) \\ y_p^{k+1} + \sum_{j=1}^p \frac{\partial f_p}{\partial y_j}(y^k)(y_j^k - y_j^{k+1}) \end{bmatrix} + h_n J_2[y^k - y^{k+1}] \quad (8)$$

Convergence leading to y^n can be interpreted here as the fixed point of that function :

$$y^{k+1} = M^{-1} \begin{bmatrix} y_A^{n-1} + h_n f_1(y^k) \\ \dots \\ y_{p-1}^{n-1} + h_n f_{p-1}(y^k) \\ f_p(y^k) \end{bmatrix} + \begin{bmatrix} \frac{\partial f_p}{\partial y_A}(y^k) y_A^k \\ \dots \\ \frac{\partial f_p}{\partial y_{p-1}}(y^k) y_{p-1}^k \\ \sum_{j=1}^p \frac{\partial f_p}{\partial y_j}(y^k) y_j^k \end{bmatrix} + h_n J_2 y^k \quad (9)$$

→ Strong assumptions on f_p and its derivatives to ensure contraction !
Whereas, in the purely differential case (f_p and the p^{th} line disappear) :

$$y^{k+1} = (Id_p + h_n J_2)^{-1} \begin{bmatrix} y_A^{n-1} + h_n f_1(y^k) \\ \dots \\ y_{p-1}^{n-1} + h_n f_{p-1}(y^k) \end{bmatrix} + h_n J_2 y^k \quad (10)$$

→ $M = Id_p + h_n J_2$ can be made contractive for a certain h_n , in the vicinity of the Newton search

In the general context of real chemical equilibria, the algebraic equations will come in many unpleasant shapes and impose (at best !) a **lower bound to the number of Newton iterations needed, independently of the value of h_n .**

→ A good rule of thumb is to give l_u a value close to that of the lower bound, which depends on the system.

→ A guess can be made from the behaviour of the system in the first timesteps : $l_u = l_1$ and $l_{max} = 3l_u$ ($\gamma_a = 1.2$ and $\gamma_d = 0.5$ for ARES and $\gamma_a = 1.5$ and $\gamma_d = 0.8$ for ARES delayed).

- ARES aims at solving a paradox : due to Stiff Nonlinear DAE, reactive transport simulations are usually very costly but it is common to have to run a transient calculation in order obtain a steady state in a robust way, although this costly transient calculation itself is not necessarily of interest.
- By allowing a relative loss of accuracy on a transient calculation, ARES may help in providing the desired result at a reduced cost, by proposing an intermediate approach between overly simple methods such as the constant timestep Euler scheme and complex high-order approaches.
- ARES is very easy to implement and to use, since it uses very few parameters.