Semi-explicit Parareal method based on convergence acceleration technique

Loïc MICHEL

May 5, 2014

Abstract

The Parareal algorithm is used to solve time-dependent problems considering multiple solvers that may work in parallel. The key feature is an initial rough approximation of the solution that is iteratively refined by the parallel solvers. We report a derivation of the Parareal method that uses a convergence acceleration technique to improve the accuracy of the solution. Our approach uses firstly an explicit ODE solver to perform the parallel computations with different time-steps and then, a decomposition of the solution into specific convergent series, based on an extrapolation method, allows to refine the precision of the solution. Our proposed method exploits basic explicit integration methods, such as for example the explicit Euler scheme, in order to preserve the simplicity of the global parallel algorithm. The first part of the paper outlines the proposed method applied to the simple explicit Euler scheme and then the derivation of the classical Parareal algorithm is discussed and illustrated with numerical examples.

1 Introduction

Despite the lack of accuracy that prevent it from being used as an ODE solver, the Euler method is known to be very easy to implement [1]. Many efficient methods have been successfully implemented to solve ODEs in a parallel way. For this purpose, the Parareal algorithm [2] introduces the possibility of parallelizing the computations requested to solve efficiently ODEs using e.g. an Euler scheme. This method uses a predictor-corrector algorithm,
applied to an implicit Euler scheme, and allows faster simulations, especially if parallel computing is available \cite{3}. Further analysis and stability proofs can be found e.g. in \cite{3} \cite{4} \cite{5} \cite{6} \cite{7}. The Parareal method uses originally coarse and fine implicit propagators to evaluate iteratively the solution of the considered ODEs, whose precision increases according to the iterations.

To extend the original properties, the proposed method uses, after the first iteration, an explicit Euler solver for both coarse and fine propagators in order to avoid difficulties related to matrix inversion and thus to keep the advantages of the simplest Euler scheme. The fine propagator recomputes iteratively and corrects the solution using different time-steps, that can be extrapolated using a convergence acceleration method in order to reach a more accurate solution. The use of extrapolation algorithms has been already considered to improve the accuracy of the solution of ODEs considering the extrapolation of the expansion of the error relating to the powers of the time-step in the case of explicit and implicit numerical integration methods (e.g. \cite{8} \cite{9}). Other derivations of the Parareal method, including e.g. a symplectic derivation \cite{10}, an adjoint-based approach \cite{11}, a relaxation-based derivation \cite{12} and a multiscale approach \cite{13} have been also successfully applied.

The paper is structured as follows. Section 2 presents the outline of the method applied to the explicit Euler scheme. Section 3 gives a brief overview of the acceleration algorithm used to increase the precision of the solution of ODEs regarding the application to the explicit Euler scheme. Section 4 reviews the standard Parareal algorithm and describes the proposed derivation including some illustrations with numerical simulations. Some concluding remarks may be found in Section 5.

2 Outline of the method

In this section, the proposed method is applied to the simple explicit Euler scheme before being applied to the Parareal algorithm using the same principle.

Consider an ordinary differential equation, eventually non-linear, such as:

\[
\frac{dy(t)}{dt} = A(t)y(t) + B(t)u(t), \quad t \in [0, T_f], \quad y(0) = y_0
\]

The quantities $u$ and $y$ represent respectively the input and the solution of (1), which can be a scalar or a vector. To solve (1) and to compute an initial
solution over $[0, T_f]$, we use the forward Euler method, for which we assume initially that the time-step $h_0$ is large and ensure the stability. The equation \[(1)\] is rewritten in the discrete time-domain:

\[
y^0(t_{k_0+1}) - y^0(t_{k_0}) = A_{k_0} y^0(t_{k_0}) + B_{k_0} u_{k_0}
\]

whose solution $y^0$ verifies ($I$ is the identity matrix whose size is compatible with the size of $A$):

\[
y^0(t_{k_0+1}) = (I + A_{k_0} h_0) y^0(t_{k_0}) + B_{k_0} h_{k_0} u_{k_0}
\]

The coefficient $h_0$ is the initial time-step such as the solution $y^0$ is calculated at each instant $t_{k_0} = k_0 h_0, k_0 \in \mathcal{K}_0 \subset \mathbb{N}$. $A_{k_0}$ and $B_{k_0}$ are connected to $A(t)$ and $B(t)$ and are described by specific relationships that depend on the smoothness of $A(t)$ and $B(t)$. In this paper, we consider $A$ and $B$ that do not depend on the time. We call $I + A_{k_0} h_0$, the dynamic matrix of the explicit Euler scheme. The value $y_0$ is considered as the initial condition. To solve accurately the explicit Euler scheme, one performs a first resolution using a high time-step $h_0$. Then, for each instant $k_0 h_0$ of the initial solution, the initial time-step $h_0$ is divided by a finite number $i \in \mathcal{I} \subset \mathbb{N}$ of subdivisions $\delta_i$ such as $h_i = h_0 / \delta_i$. Some power computations of $(I + A_{k_0} h_0)$ give a sequence for which, the limit, and therefore, a good estimation of the true solution, is deduced using a convergence accelerator. We denote:

\[
\Psi^0 = (y_0, y^0(t_1), y^0(t_2), \ldots, y^0(t_{k_0}), \ldots, y^0(T_f))
\]

the series that contains the points of the calculated solution (with the time-step $h_0$) at each instant $t_{k_0} = k_0 h_0$. Consider now a smaller time-step $h_1 = h_0 / \delta_1 < h_0$, with $\delta_1 > 1$, for which we obtain the series of the calculated solutions:

\[
\Psi^1 = (y_0, y^1(t_1), y^1(t_2), \ldots, y^1(t_{k_0}), \ldots, y^1(T_f))
\]

at each instant $t_{k_1}^1 = k_1 h_1 = k_1 h_0 / \delta_1, k_1 \in \mathcal{K}_1 \subseteq \mathcal{K}_0 \subset \mathbb{N}$. We have thus $t_{k_0 \delta_1}^1 = t_{k_0}^0$. Therefore, by induction, we can deduce that for any time-step $h_i = h_0 / \delta_i < h_{i-1}$ with $\delta_i > \delta_{i-1} > 1$, the series of the calculated solutions:

\[
\Psi^i = (y_0, y^i(t_1), y^i(t_2), \ldots, y^i(t_{k_0 \delta_1}), \ldots, y^i(T_f))
\]
at each instant \( t^i_k = k_i h_i = k_i h_0 / \delta_i, \) \( k_i \in K_i \subseteq K_0 \subset \mathbb{N} \) and thus \( t^i_{k_0 \delta_i} = t^0_{k_0} \). Similarly, we denote by \( \Psi^T \) the series of the points that correspond to the "true" solution:

\[
\Psi^T = (y_0, y^T(t^0_1), y^T(t^0_2), \ldots y^T(t^0_{k_0}), \ldots, y^T(T_f))
\]

at each instant \( t^0_{k_0} \). Note that since \( h_0 \) is the initial time-step and thus the "reference" time-step, \( y^T \) is computed at the same instants \( t^0_{k_0} \). Therefore, at each instant \( t^0_{k_0} \), the solution is described as a finite series composed of terms that are computed only from the power computations of the dynamic matrix (the time-step scaling factor \( \delta_i \) allows computing each series \( \Psi^i \) at the same instants \( t^0_{k_0} \) as taken for the solution \( \Psi^0 \)). At each instant \( t^0_{k_0} \), the solution (3) is described as \( \Omega_{k_0} \)-series (composed of the underlined terms of each \( \Psi^i, i \in I \)):

\[
\Omega_{k_0} = (y^0(t^0_{k_0}), y^1(t^1_{k_0 \delta_1}), \ldots, y^i(t^i_{k_0 \delta_i}), \ldots)
\]

with:

\[
\lim_{i \to \infty} y^i(t^i_{k_0 \delta_i}) = \Omega^i_{k_0} \approx y^T(t_{k_0})
\]

To obtain an accurate estimation of the limit \( \Omega^i_{k_0} \), we will describe the \( \varepsilon \)-algorithm, which is a convergence accelerator algorithm, whose purpose is to estimate the limit \( \Omega^i_{k_0} \) from only a few terms of \( \Omega_{k_0} \).

3 Shanks transform of numerical series

Consider \( (S_n)_{n \in \mathbb{N}} \) (also simply \( S_n \)), a real series that converges to a limit \( S^\text{lim} \) for \( n > n_{\text{lim}} \). To accelerate the convergence, one defines the transformation \( \Phi : \mathbb{R} \to \mathbb{R} \), such as \( T_n = \Phi(S_n) \), whose limit is \( T_\infty \), implies that \( T_\infty - S^\text{lim} = o(S_n - S^\text{lim}) \). In other words, we define the transformation \( \Phi \), such as the transformed series \( T_n \) from the series \( S_n \) reaches its limit \( S^\text{lim} \) faster than \( S_n \). Among the different methods that has been established to accelerate the convergence of a series [14] [15], we consider the \( \varepsilon \)-algorithm [16] [17] [18] based on the Shanks transforms [19] [20] [21] [22], that is reputed to be one the most powerful algorithm [23] to accelerate nonlinear series.

3.1 The \( \varepsilon \)-algorithm

Consider a series \( S_n \), for which we aim to estimate the limit \( S^\text{lim} \) taking into that \( S_n \) is the \( n \)th element of the series. From a few terms of \( S_n \), the limit
$S^\text{lim}$ can be extrapolated using the following algorithm \[16\]:

$$
\begin{aligned}
\varepsilon_{-1}^{(n)} &= 0, \\
\varepsilon_0^{(n)} &= S_n, \\
\varepsilon_{k+1}^{(n)} &= \varepsilon_{k-1}^{(n)} + (\varepsilon_k^{(n)} - \varepsilon_{k-1}^{(n)})^{-1}, \\
\end{aligned}
$$

where $S_n$ could be a one-dimension series or a multidimensional series. The $\varepsilon$-algorithm can be described as a function $S^*_n = S^*_\varepsilon(k, n, S_n)$ that extrapolates $S_n$ and gives a new series $S^*_n$ that converges faster to $S^\text{lim}$ (the limit $S^\text{lim}$ should be reached after a few terms of $S^*_n$), where $S_n$ is the series for which we aim to estimate the limit $S^\text{lim}$. The parameters of the extrapolation function $k$ and $n$ are respectively the index of $S^\varepsilon$ ($k$ is also the order of the associated Shanks transform) and the initial index of $S_n$. Table 1 gives the number of terms of $S_n$ and the corresponding number of computations (“nb. of cmp.”) that are requested in order to compute $S^\varepsilon(k, n, S_n)$. Only even $k$ number are considered \[17\] and any index $n$ from $S_n$ may be considered $((k, n) \in \mathbb{N})$. We can deduce that the estimation of the limit $S^\text{lim}$ is more accurate when $k$ and $n$ are sufficiently high (depending on the convergence rate of $S_n$). However, higher index $k$ involve much more computations that may decrease the overall efficiency of the method. As a result, a compromise has to be found between $k$ and $n$ in order to get the most accurate $S^\text{lim}$ with a few terms of $S_n$.

3.2 Application to the explicit Euler scheme

We present the general strategy applied to the explicit Euler scheme that will be transposed to the explicit-Euler based Parareal algorithm in the next section. To provide a ”generic” form of the explicit Euler induction, we assume that $A, B$ and $u$ are constants using the ”initial” time-step $h_0$. From
the first recursive terms of (3):
\[ y^0(t_0^1) = (I + h_0 A)y^0(t_0) + h_0 Bu \]
\[ y^0(t_0^2) = (I + h_0 A)y^0(t_0^1) + h_0 Bu \]
\[ \vdots \]
\[ y^0(t_{k_0}) = (I + h_0 A)y^0(t_{k_0-1}) + h_0 Bu \]
we obtain the following definition of the explicit Euler scheme:
\[ y^0(t_{k_0}) = (I + h_0 A)^{k_0} y^0(t_0) + \sum_{j=0}^{k_0-1} (I + A h_0)^j h_0 Bu, \quad k_0 \in \mathcal{K}_0 \quad (7) \]

According to the Section 2, we introduce the parameter \( \delta_i \), that divides the time-step \( h_0 \) such as \( h_i = h_0 / \delta_i \), and the time-iteration \( k_i \), as a subdivision of \( k_0 \). Therefore, (7) is rewritten at each same instants:
\[ y^i(t_{k_0}) = (I + A h_0)^{k_0} y^0(t_0) + \sum_{j=0}^{k_0-1} (I + A h_0)^j h_0 Bu, \quad k_i \in \mathcal{K}_i \quad (8) \]

For all \( \delta_i \), the corresponding \( \Omega_{k_0} \) series “collects” each value of \( y^i(t_{k_0}) \) and one retrieve the series (4):
\[ \Omega_{k_0} = (y^0(t_{k_0}), y^1(t_{k_0}), \ldots, y^i(t_{k_0}), \ldots) \]
with the limit \( \Omega_{k_0}^{lim} \) given by (5). Then, for each time-iteration \( k_0 \), we define the accelerated series \( \Omega_{k_0}^* \) such as: \( \Omega_{k_0}^* = S_\varepsilon(k, n, \Omega_{k_0}) \). The estimated limit is noted \( \Omega_{k_0}^{* lim} \) and is given by the last term(s) of \( \Omega_{k_0}^* \).

4 Application to the Parareal method

4.1 Review of the Parareal method approach

The Parareal algorithm is a parallel method that computes numerical solutions of general systems of ordinary differential equations (ODEs) of the form (1):
\[ U' = f(U) \quad U(0) = U_0 \quad (9) \]
over the time interval \( t \in [0, T] \) decomposed into \( N \) subdivisions such as:
\( t = t_0, t_1, \ldots, t_j, \ldots, t_N \), where \( j \in J \subset \mathbb{N} \). Two propagation operators are defined:
• the implicit operator "coarse" \( G_i(t_{j+1}, t_j, U_j, h_g) \), where \( h_g \) is the time-step, provides a rough approximation to \( U(t_{j+1}) \) of the solution of (9) with initial condition \( U(t_j) = U_j \);

• the implicit operator "fine" \( F_i(t_{j+1}, t_j, U_j, h_f) \), where \( h_f < h_g \) is the time-step, provides a more accurate approximation of \( U(t_{j+1}) \) of the solution of (9) with initial condition \( U(t_j) = U_j \).

The algorithm starts with an initial approximation \( U^0_j, j = 0, 1, \ldots, N \) at times \( t_0, t_1, \ldots, t_N \) using the \( G \) operator. The solution of (9) given by the \( G \) solver is improved by the \( F \) solver. The purpose of the Parareal method is to integrate (9) using the \( G \) solver with a high time-step and then to improve iteratively, via the \( F \) solver, the solution at the points calculated by the \( G \) solver with a lower time-step \( h_f < h_g \). To correct iteratively the values of the solution of (9) taking into account both solutions of \( G \) and \( F \) solvers, (10) allows evaluating the modified points of the \( G \) solver according to the iteration \( k \geq 1 \):

\[
U^{k+1}_{j+1} = G_i(t_{j+1}, t_j, U^{k+1}_j, h_g) + F_i(t_{j+1}, t_j, U^k_j, h_f) - G_i(t_{j+1}, t_j, U^k_j, h_g) \tag{10}
\]

where \( j \) designates the \( j \)th time subdivision of the solution of (9) and \( k \) the iteration during the Parareal process. The accuracy of the final solution of (9) when \( k \to \infty \) depends on the accuracy of \( F \) [6].

4.2 Semi-explicit derivation of the Parareal method

Our proposed approach follows the classical scheme of the Parareal method described in the previous subsection. To ensure the stability of the global algorithm, the rough solution at the first iteration \( k = 1 \) is performed by an implicit operator \( G_i(t_{j+1}, t_j, U_j, h_g) \). Then, for \( k > 1 \),

• the explicit operator "coarse" \( G_e(t_{j+1}, t_j, U_j, h_g) \), where \( h_g \) is the time-step, provides a rough approximation to \( U(t_{j+1}) \) of the solution of (9) with initial condition \( U(t_j) = U_j \);

• the explicit operator "fine" \( F_e(t_{j+1}, t_j, U_j, h_g/\delta_k) \), where \( h_g/\delta_k \) is the time-step, provides a more accurate approximation of \( U(t_{j+1}) \) of the solution of (9) with initial condition \( U(t_j) = U_j \).
The time-step of the $F$ operator is reduced according to the $k$ iteration. Therefore, the evolution of the precision of the corrected values of the solution of (9) has the same behavior than the evolution of the precision in the case of the simple explicit Euler method. Expressions (11) and (12) allows evaluating the modified points of the $G$ solver according to the iteration $k \geq 1$:

- if $k = 1$, then:
  $$U^{k+1}_{j+1} = G_e(t_{j+1}, t_j, U^{k+1}_{j}, h_g) + F_e(t_{j+1}, t_j, U^k_{j}, h_f/\delta_k) \quad \text{-- (11)}$$

- if $k > 1$, then:
  $$U^{k+1}_{j+1} = G_e(t_{j+1}, t_j, U^{k+1}_{j}, h_g) + F_e(t_{j+1}, t_j, U^k_{j}, h_f/\delta_k) \quad \text{-- (12)}$$

At each instant $t_j$, the solution is described as $\Omega_j$-series constituted from each modified points $U^k_j$ such as:

$$\Omega_j = (U^2_j, U^3_j, \ldots U^k_j, U^{k+1}_j, \ldots) \quad \text{-- (13)}$$

with the limit $\Omega^\lim_j$ given by (5). Besides the efficiency of parallelizing the operations, the main advantages of this technique is to reduce the error propagated by the $F_e$ and $G_e$ solvers thanks to the combination of the algorithms (11) and (12) and the solvers that manage the resolution over smaller time intervals $[t_j, t_{j+1}] \subset [0, T]$ for all $j \in J$.

To evaluate the limit $\Omega^\lim_j$, we consider the application $\Omega^*_j = S_e(k, n, \Omega_j + S^n_b)$ where we take $k = 4$ and $n = 2$ such as the calculation of the extrapolation is performed considering a very few terms of $\Omega_j$ (according to the Table 4 if $k = 6$, much more computations are requested), and $S^n_b$ is an alternating series that allows increasing the precision of the estimated limit [23]. This series is of the form:

$$S^n_b = S^n_0 + (-1)^n \sum_{j=1}^{n} \frac{1}{(n + 1)^q} \quad \text{-- (14)}$$

where $S^n_0$ is the initial term of the series. Since $S^n_b$ is a convergent series that has a "resonant" transient, $q$ is a real positive number that characterizes the damping of the resonance. The coupling between the alternating series $S^n_b$ and the series $\Omega_j$ to accelerate, highlights an important point that concerns
The resolution of the \( \Omega_j \) series. For a given \( j \in J \), since only four terms of \( \Omega_j \) are requested to compute the extrapolation, an additional parameter of the extrapolation function is the distance between each term of \( \Omega_j \). We define the resolution of the series \( \Omega_j \) as the distance \( \delta \)-distance defined by \(|\delta_k - \delta_{k+1}|\). For all \( k \), we define \( \delta_1 \) and \( \delta_2 \) such as: \( \delta_1 < |\delta_k - \delta_{k+1}| < \delta_2 \) with \( \delta_1 < \delta_2 \) and \((\delta_1, \delta_2) \in \mathbb{R}^+\).

In the next subsection, a complete procedure summarizes the computations needed to solve (9) and calculate all limits \( \Omega_j^{\lim}, j \in J \).

### 4.3 Optimization of the extrapolation of \( \Omega_j^* \)

To improve the use of the Wynn algorithm, an optimization procedure is considered. The goal is to adjust the damping of \( S^b_0 \), by adjusting the coefficient \( q \), in order to reduce the error \( \Omega_j^*, j \in J \). This technique implies theoretically to get a large amount of terms of the \( \Omega_j \)-series for all \( j \in J \) in order to have a good estimation of the limits \( \Omega_j^{\lim} \), with which \( \Omega_j^{*\lim} \) will be compared. However, from a practical side, we consider only \( \Omega_1 \) for the optimization and the resulting \( q \) value, noted \( q_{opt} \), will be propagated through the other series \( \Omega_j, j > 1 \). The following steps, that initializes the optimization procedure, can be considered.

1. To obtain \( \Omega_1^{\lim} \), the explicit Euler scheme is directly applied on the time-interval \([0, t_1]\) with a very low time-step.

2. To obtain the optimized \( q_{opt} \), the first four-term of the \( \Omega_1 \) series are considered. The resulting \( q_{opt} \) is hold for the rest of the series \( \Omega_j, j > 1 \). Therefore, the rate of convergence of the \( \Omega_j \) series is assumed to be constant for all \( j \in J \).

Consider the 3-variable general optimization problem (valid for all \( j \in J \)):

\[
\min_{q > 0, \delta_1 > 0, \delta_2 > 0} \left\{ \rho S_\varepsilon \left( 4, 2, \left( \Omega_j + S^0_0 + (-1)^n \sum_{j=1}^n \frac{1}{(n+1)^q} \right) \right) - y^T(t_1^0) \right\}
\]

subjected to

\[
\delta_1 < |\delta_k - \delta_{k+1}| < \delta_2, \quad \delta_1 < \delta_2,
\]

where \( \rho \in \mathbb{R}^{**} \) is a scaling factor whose purpose is to scale \( \Omega_1 \) in order to be of the same range than the auxiliary series. The following complete
algorithmic procedure enumerates the steps in order to evaluate the limits \( \Omega_j^{\text{lim}} \) for all \( j \in \mathcal{J} \).

1. Parareal computation The proposed derivation of the Parareal method solves the ODEs, using \( G_i, G_e \) and \( F_e \) solvers and applying (11) and (12) as described in the previous subsection. As a result, the series \( \Omega_j \) are computed for all \( j \in \mathcal{J} \).

2. Initialization step From the \( \Omega_j \)-series, the initialization consists in calibrating the convergence accelerator, by determining firstly the \( \Omega_1^{\text{lim}} \) using an explicit Euler scheme, and then the \( q_{opt} \) value for which the global error \( \Omega_1^{\varepsilon} \) is minimized using (15). Remind that a small index \( k \) is considered to apply the Wynn algorithm that limits the number of requested terms from \( \Omega_j \) and the number of computations.

3. Final acceleration It follows that each approximated limit \( \Omega_j^{\text{lim}}, j > 1 \) is calculated using the expression (16).

\[
\Omega_j^{\text{lim}} = S_{\varepsilon} \left( 4, 2, \left( \Omega_j + S_0^0 + (-1)^n \sum_{j=1}^{n} \frac{1}{(n+1)^{q_{opt}}} \right) \right)
\]

The series \( \Omega^{\text{lim}} \) composed of the \( \Omega_j^{\text{lim}} \) terms for \( j \in \mathcal{J} \) such as:

\[
\Omega^{\text{lim}} = (\Omega_1^{\text{lim}}, \Omega_2^{\text{lim}}, \Omega_3^{\text{lim}}, \cdots)
\]

constitutes the solution of (9), calculated by the Parareal method at the instants \( t = t_1, t_2, t_3, \cdots \) with only a few iterations, and extrapolated using the Wynn algorithm.

**Illustrative example:** Consider in this paper, a linear second order system \( \Sigma \), described by a state-space representation:

\[
\Sigma := \begin{pmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{pmatrix} = \begin{pmatrix}
-1 & 5 \\
-5 & -1
\end{pmatrix} \begin{pmatrix}
x_1 \\
x_2
\end{pmatrix} + \begin{pmatrix}
0 \\
1
\end{pmatrix} u \quad x_1(0) = 0, x_2(0) = 1
\]

considering \( u = 10 \). Figure 1 presents the different solutions of \( \Sigma \), relating to different time-step in the phase-space \( x_1 - x_2 \) by applying the explicit Euler scheme. The ”true solution” is the exact solution of \( \Sigma \). Starting from a time-step \( h_0 \), that gives a very inaccurate solution, the solution is computed
considering some subdivisions $\delta_i$, $i = 1 \cdots 5$. At each instant $t_{k_0}$, given some terms of the series $\Omega_{k_0}$, the acceleration algorithm allows having an accurate estimation of $\Omega_{k_0}^{lim}$.

Figure 1: True solution of $\Sigma$ plotted in the phase-space in comparison with explicit Euler scheme with different time-steps $h_i$ (corresponding to the $\Psi_i$ series).

In the case of the explicit Euler scheme, Figure 2 depicts the typical logarithmic evolution [24] of the error $\Omega_{k_0} = |\Omega_{k_0} - y^T(t_{k_0}^0)|$ series according to $\delta_i$ for $h_0 = 0.1$ s and $k_0 = 1 \cdots 5$. The application of the proposed strategy to evaluate the limit of each $\Omega_{k_0}$ series does not provide any significant improvement taking into account that an accurate (asymptotic) precision can be obtained usually from $\delta_i \geq 50$ considering small $k_0$. Indeed, the global error increases according to $k_0$ due to the error propagation throughout the integration of (18) over the time.

The Parareal technique is now applied to solve (18). Figure 3 presents the evolution of each $\Omega_j^e = |\Omega_j - y^T(t_j^0)|$ series according to $\delta_j$ for $h_0 = 0.1$ s and $j = 1 \cdots 9$.

To verify the main assumption for which $q$ is kept constant for all $\Omega_j$ (§4.3, item 2), we search the optimal $q_{opt}$ by applying (15) to each series $\Omega_j$. A meta-heuristic optimization procedure, like the simulated annealing method [25], is applied. The following tables 2 3 4 5 6 7 show the result of the search of the $q_{opt}$ values considering some different fixed $\delta^\Omega$-distance values and fixed $\delta_j$ values (the time-step divisor $h_0/\delta_i$) in the Euler scheme. For all
cases, the goal is to find $q_{opt}$ such as both $|\Omega_j^{*lim} - \Omega_j^{lim}|$ and $|\Omega_j^{*lim} - y(t_0^j)|$ are minimized separately.

The estimated limit $\Omega_j^{*lim}$ is compared to the limit $\Omega_j^{lim}$ given by the asymptotic behavior of $\Omega_j$ (via the Euler scheme) and the true solution $y(t_0^j)$. The results show that a better estimation of the limit $\Omega_j^{*lim}$ is obviously obtained for a high value of $\delta_i$ but in other hand, the corresponding $\delta^{\Omega}$-
Table 2: Explicit Euler scheme with $\delta_j = 500$ and $\delta$-distance $= 0.5$

\begin{tabular}{c|c|c}
  $j$ & $q_{\text{opt}} \rightarrow |\Omega_j^{\text{true}} - \Omega_j^{\text{est}}| \times 10^3$ & $q_{\text{opt}} \rightarrow |\Omega_j^{\text{true}} - y(t_j^*)| \times 10^3$ \\
  \hline
  1 & 0.0022 & 9.5995 & 0.0022 & 26.4484 \\
  2 & 0.0022 & 9.7795 & 0.0022 & 7.7027 \\
  3 & 0.0022 & 9.5684 & 0.0022 & 9.1885 \\
  4 & 0.0022 & 219.3907 & 0.0022 & 524.8596 \\
  5 & 0.0022 & 65.2311 & 0.0022 & 9.1266 \\
  6 & 0.0022 & 7.9504 & 0.0022 & 8.0144 \\
  7 & 0.0022 & 600.3367 & 0.0022 & 7.1330 \\
  8 & 0.0022 & 9.3852 & 0.0022 & 122.9864 \\
  9 & 0.0022 & 189.1382 & 0.0022 & 8.8399 \\
\end{tabular}

Table 3: Explicit Euler scheme with $\delta_j = 100$ and $\delta$-distance $= 0.1$

\begin{tabular}{c|c|c|c|c}
  $j$ & $q_{\text{opt}} \rightarrow |\Omega_j^{\text{true}} - \Omega_j^{\text{est}}| \times 10^4$ & $q_{\text{opt}} \rightarrow |\Omega_j^{\text{true}} - y(t_j^*)| \times 10^4$ \\
  \hline
  1 & 0.0035 & 1.1811 & $\leq 10^{-10}$ & 5.0441 \\
  2 & $\leq 10^{-10}$ & 5.0403 & 0.0035 & 5.0051 \\
  3 & $\leq 10^{-10}$ & 5.0403 & 0.0035 & 0.0040 \\
  4 & 0.0035 & 0.0005 & $\leq 10^{-10}$ & 5.0441 \\
  5 & $\leq 10^{-10}$ & 5.0403 & 0.0035 & 0.5065 \\
  6 & $\leq 10^{-10}$ & 5.0403 & $\leq 10^{-10}$ & 5.0441 \\
  7 & 0.0035 & 5.0359 & 0.0035 & 0.0515 \\
  8 & $\leq 10^{-10}$ & 5.0403 & $\leq 10^{-10}$ & 5.0441 \\
  9 & $\leq 10^{-10}$ & 5.0403 & $\leq 10^{-10}$ & 5.0441 \\
\end{tabular}

distance has to be properly bounded. In particular, we deduce a rough approximation of the lower bound such as $\delta_1 = 0.5$. Table 6 presents the results for which the best estimations of the limits have been obtained. These results confirm that the same $q_{\text{opt}}$ can be used to treat all $\Omega_j, j \in J$.

Remark In the case where the system (9) has to be solved over a large time-interval, the initialization step can be applied periodically to ”refresh” the optimized parameters and thus to ensure good performances.
5 Conclusion

We presented a derivation of the Parareal algorithm scheme that uses a convergence accelerator algorithm in order to improve the precision of the resulting solution at each instant. The main improvements concern the use of an explicit solver for the $G$ and $F$ operators and the use of an acceleration technique, like the Wynn algorithm, to improve the convergence. The proposed method has the following features:

- The computations are based essentially on the power computation of the dynamic matrix $(I + A_{k_0}h_i)$.
- Since the power computations involve several time-steps (the initial
Table 6: Explicit Euler scheme with $\delta_j = 100$ and $\delta$-distance = 1

| $j$ | $q_{opt} \rightarrow |\Omega_j^{lim} - \Omega_j^{opt}| \times 10^4$ | $q_{opt} \rightarrow |\Omega_j^{lim} - y(t_j^m)| \times 10^4$ |
|-----|-------------------------------------------------|-------------------------------------------------|
| 1   | $\leq 10^{-10}$                                 | $4.8732$                                         |
| 2   | $\leq 10^{-10}$                                 | $4.8732$                                         |
| 3   | $\leq 10^{-10}$                                 | $4.8732$                                         |
| 4   | $\leq 10^{-10}$                                 | $4.8732$                                         |
| 5   | $\leq 10^{-10}$                                 | $4.8732$                                         |
| 6   | $\leq 10^{-10}$                                 | $4.8732$                                         |
| 7   | $\leq 10^{-10}$                                 | $4.8732$                                         |
| 8   | $\leq 10^{-10}$                                 | $4.8732$                                         |
| 9   | $\leq 10^{-10}$                                 | $4.8732$                                         |

Table 7: Explicit Euler scheme with $\delta_j = 50$ and $\delta$-distance = 0.5

| $j$ | $q_{opt} \rightarrow |\Omega_j^{lim} - \Omega_j^{opt}| \times 10^4$ | $q_{opt} \rightarrow |\Omega_j^{lim} - y(t_j^m)| \times 10^4$ |
|-----|-------------------------------------------------|-------------------------------------------------|
| 1   | 0.0022                                           | 8.7869                                           |
| 2   | 0.0022                                           | 60.2327                                          |
| 3   | 0.0022                                           | 8.9733                                           |
| 4   | 0.0022                                           | 18.8258                                          |
| 5   | 0.0022                                           | 8.0245                                           |
| 6   | 0.0022                                           | 9.8895                                           |
| 7   | 0.0022                                           | 787.8841                                         |
| 8   | 0.0022                                           | 8.3683                                           |
| 9   | 0.0022                                           | 9.3224                                           |

- Multidimensional ODEs can be considered.

Future investigations include the improvement of the series accelerator algorithm relating to the alternating series, the development of an adaptive scheme (e.g. a variable time-step scheme), the development of a symplectic approach and the extension of the method to solve DAEs / stiff systems. Moreover, the association and implementation with existing innovative methods are also of interest. A demonstration code is available upon request to the author.
References

[1] W. Y. Yang, Applied Numerical Methods Using MATLAB, Wiley, (2005).

[2] J.-L. Lions, Y. Maday, G. Turinici, Résolution d’EDP par un schéma en
temps pararéel, Comptes Rendus de l’Académie des Sciences - Series I -
Mathematics, 332, (2001).

[3] Y. Maday, G. Turinici, The parareal in time iterative solver: a further
direction to parallel implementation, Domain decomposition methods in
science and engineering, Lecture Notes in Computational Science and En-
gineering, Springer Berlin Heidelberg, 40, 441-448 (2005).

[4] Y. Maday and G. Turinici. A parareal in time procedure for the control of
partial differential equations, Comptes Rendus de l’Académie des Sciences
- Series I - Mathematics, 335, 387-392 (2002).

[5] G. Staff and E. Rønquist. Stability of the parareal algorithm, Domain
decomposition methods in science and engineering, Lecture Notes in Com-
putational Science and Engineering, Springer, 40, 449-456 (2005).

[6] S. Vandewalle, M. Gander, Analysis of the Parareal Time-Parallel Time-
Integration Method, SIAM Journal on Scientific Computing, 29, 556-578
(2007).

[7] M. Gander, S. Vandewalle, On the Superlinear and Linear Convergence of
the Parareal Algorithm, Domain Decomposition Methods in Science and
Engineering, Lecture Notes in Computational Science and Engineering,
Springer, 55, 291-298 (2007).

[8] P. Deuflhard, Recent Progress in Extrapolation Methods for Ordinary
Differential Equations, SIAM Rev., 27, 505-535 (1985).

[9] E. M. Constantinescu, A. Sandu, Extrapolated Implicit-Explicit Time
Stepping, SIAM J. Sci. Comput., 31, 4452-4477 (2010).

[10] G. Bal, Q. Wu, Symplectic Parareal, Domain Decomposition Methods
in Science and Engineering, Lecture Notes in Computational Science and
Engineering, 60, 401-408 (2008).

[11] V. Rao, A. Sandu, An Adjoint Based Implementation of the Parareal
Algorithm, Procedia Computer Science, 9, 1021-1029 (2012).
[12] J. Liu, Y.-L. Jiang, A parareal algorithm based on waveform relaxation, Mathematics and Computers in Simulation, 82, 2167-2181 (2012).

[13] F. Legoll, T. Lelievre, G. Samaey, A micro/macro parareal algorithm for singularly perturbed ordinary differential equations, SIAM Journal on Scientific Computing, 35, 1951-1986 (2013).

[14] D. A. Smith and W. F. Ford, Numerical Comparisons of Nonlinear Convergence Accelerators, Mathematics of Computation (AMS), 38, 481-499 (1982).

[15] J. Wimp, Sequence transformations and their applications. Math in Science & Engineering, Academic Press N-Y, 514, (1984).

[16] P. Wynn, On a device for computing the $em(S_n)$ transformation, American Mathematical Society, 10, 91-96 (1956).

[17] P. Wynn, On the Convergence and Stability of the Epsilon Algorithm, SIAM Journal on Numerical Analysis, 3, 91-122 (1966).

[18] P.R. Graves-Morris, D.E. Roberts, A. Salam, The epsilon algorithm and related topics, Journal of Computational and Applied Mathematics, 122, 51-80 (2000).

[19] D. Shanks, Non-linear transformation of divergent and slowly convergent sequences, Journal of Mathematics and Physics, 34, 1-42 (1955).

[20] S. Singh, R. Singh, On the use of Shank's transform to accelerate the summation of slowly converging series, IEEE Transactions on Microwave Theory and Techniques, 39, 608-610 (1991).

[21] A. Sidi, Survey of numerical stability issues in convergence acceleration, Applied Numerical Mathematics, 60 (2010).

[22] A. Sidi, Acceleration of convergence of general linear sequences by the Shanks transformation, J. Numerische Mathematik, Springer-Verlag, 119, (2011).

[23] C. Brezinski, Accélération de la Convergence en Analyse Numérique. Lecture Notes in Mathematics, Springer-Verlag, Heidelberg, 584, (1977).
[24] J. C. Butcher, Numerical Methods for Ordinary Differential Equations, J. Wiley, Chichester, (2008).

[25] V. Granville, M. Krivanek, J.-P. Rasson, Simulated annealing: A proof of convergence, IEEE Transactions on Pattern Analysis and Machine Intelligence, 16 652-656 (1994).