De Vuyst *Adv. Model. and Simul. in Eng. Sci.(2016)3:8*
DOI 10.1186/s40323-016-0063-y

Advanced Modeling and Simulation in Engineering Sciences
a SpringerOpen Journal

RESEARCH ARTICLE Open Access

Efficient solvers for time-dependent problems: a review of IMEX, LATIN, PARAEXP and PARAREAL algorithms for heat-type problems with potential use of approximate exponential integrators and reduced-order models

Florian De Vuyst

*Correspondence:
devuyst@cmla.ens-cachan.fr
CMLA, ENS Cachan Université Paris-Saclay, CNRS, 94235 Cachan, France

Abstract

In this paper, we introduce and comment some recent efficient solvers for time dependent partial differential or ordinary differential problems, considering both linear and nonlinear cases. Here "efficient" may have different meanings, for instance a computational complexity which is better than standard time advance schemes as well as a strong parallel efficiency, especially parallel-in-time capability. More than a review, we will try to show the close links between the different approaches and set up a general framework that will allow us to combine the different approaches for more efficiency. As a complementary aspect, we will also discuss ways to include reduced-order models and fast approximate exponential integrators as fast global solvers. For developments and discussion, we will mainly focus on the heat equation, in both linear and nonlinear form.

Keywords: IMEX, LATIN, PARAEXP, PARAREAL, Performance, Reduced order model

Background

This paper deals with efficient numerical approaches to solve time-dependent problems, possibly including parallel-in-time sub-domain decomposition and making help of coarse reduced-order model solvers. As a typical problem of discussion, we will consider the classical heat equation: let Ω be a bounded domain in $\mathbb{R}^m$, $m \in \{2, 3\}$ with a Lipschitz-continuous boundary. Let κ be a positive constant. Consider $T > 0$, $u^0 \in H_0^1(\Omega)$ and $f \in L^2((0, T), L^2(\Omega))$. The linear heat problem with u^0 an initial value, homogeneous boundary conditions, for time t in the interval $[0, T]$ reads

$$\begin{cases} \partial_t u - \nabla \cdot (\kappa \nabla u) = f & \text{in } \Omega \times [0, T], \\ u(., t) = 0 & \text{on } \partial\Omega \times [0, T], \\ u(., 0) = u^0 & \text{in } \Omega. \end{cases} \tag{1}$$

The problem (1) has a unique solution u in $L^2((0, T), H_0^1(\Omega))$. Semi-discretizing the problem (1) in space (method of lines) will classically lead to a high-dimensional ordinary differential problem set in $\mathbb{R}^d$ with generally large discrete dimension d. For simplicity, we will assume that the semi-discrete problem is written

$$\begin{cases} \dot{\boldsymbol{u}} + A\boldsymbol{u} = \boldsymbol{f} & \text{in } [0, T], \\ \boldsymbol{u}(0) = \boldsymbol{u}^0, \end{cases} \tag{2}$$

with $\boldsymbol{u}^0 \in \mathbb{R}^d, \boldsymbol{f} \in L^2((0,T), \mathbb{R}^d)$ and $A \in \mathscr{M}_d(\mathbb{R})$ typically symmetric positive definite, with a sparse structure. In this paper we will also consider nonlinear versions of the heat problem with a thermal conductivity coefficient $\kappa(u)$ depending on u itself. We will assume that there exists a constant $\underline{\kappa} > 0$ and a constant $\overline{\kappa} > 0$ such that

$$\underline{\kappa} \le \kappa(u) \le \overline{\kappa} \quad \forall u.$$

The nonlinear heat problem reads

$$\begin{cases} \partial_t u - \nabla \cdot (\kappa(u)\nabla u) = f & \text{in } \Omega \times [0, T], \\ u(.,t) = 0 & \text{on } \partial\Omega \times [0, T], \\ u(.,0) = u^0 & \text{in } \Omega \end{cases} \tag{3}$$

and we will assume that its semi-discretized form reads

$$\begin{cases} \dot{\boldsymbol{u}} + A(\boldsymbol{u})\,\boldsymbol{u} = \boldsymbol{f} & \text{in } [0, T], \\ \boldsymbol{u}(0) = \boldsymbol{u}^0 \end{cases}$$

with $A(\boldsymbol{u})$ sparse, symmetric positive definite for any $\boldsymbol{u}$, uniformly bounded. Let us now consider time discretization. Usually, time advance schemes for such kind of problems are chosen implicit or semi-implicit for stability purposes. As an exemple, the pure explicit Euler time advance scheme

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} - \nabla \cdot (\kappa(\boldsymbol{u}^n)\nabla \boldsymbol{u}^n) = f$$

where $\boldsymbol{u}^n \simeq \boldsymbol{u}(.,t^n)$, $t^{n+1} = t^n + \Delta t$ has a too restrictive numerical stability domain with typically $\Delta t = O(h^2)$, h being representative of the space step size. Semi-implicit linear schemes in the form

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} - \nabla \cdot (\kappa(\boldsymbol{u}^n)\nabla \boldsymbol{u}^{n+1}) = f$$

show a far better stability domain but require the update of the stiffness matrix with the solution of a large linear sparse system at each time step. Finally, full implicit schemes

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} - \nabla \cdot (\kappa(\boldsymbol{u}^{n+1})\nabla \boldsymbol{u}^{n+1}) = f$$

provide strong numerical stability but require fixed-point (Newton or quasi-Newton) algorithms for their numerical solution, what becomes computationally time-consuming.

This paper gives an overview of recent alternative time advance schemes with interesting algorithmic features, including the possibility of parallel computations. First for linear problems, we will introduce the PARAEXP algorithm based on a superposition principle for achieving parallel-in-time computation. For nonlinear problems, the iterative LATIN method is a kind of splitting approach by alternating global linear solutions and local nonlinear projections. We will then discuss more general fixed point algorithms with a special focus on Newton and quasi-Newton methods, separation of linear terms and nonlinear residuals in an implicit-explicit discretization strategy, then time sub-domain decomposition and parallel-in-time computing involving coarse global and fine local propagators in the PARAREAL method.

The PARAEXP algorithm

Numerical methods allowing parallelization in the time direction have been thought since a long time (see Nievergelt [20] in 1964) and have known great developments particularly in the last decade because of today's growing HPC platforms. Among time-parallel solvers, the PARAEXP algorithm introduced by Gander and Güttel [10] in 2013 is dedicated to linear ordinary differential problems, that is problems in the form

$$\begin{cases} \dot{\boldsymbol{u}} + A\boldsymbol{u} = \boldsymbol{f}(t), \quad t \in [0, T], \\ \boldsymbol{u}(0) = \boldsymbol{u}^0 \end{cases} \tag{4}$$

especially when $\boldsymbol{f}(t)$ is varying fastly in time. Problem (4) has a solution written in integral form thanks to the variation-of-constant formula:

$$\boldsymbol{u}(t) = \exp(-tA)\boldsymbol{u}^0 + \int_0^t \exp(-(t-\tau)A)\boldsymbol{f}(\tau)\, d\tau. \tag{5}$$

If we want to take advantage of (5) for deriving a numerical computational method, in particular we need a high-order quadrature formula of the integral term. If the $\boldsymbol{f}(t)$ are fast varying source terms, quadrature may become irrelevant from the accuracy point of view. Gander and Güttel rather propose to split the problem over p sub-domains in time and use a superposition principle based on independent problems set onto different time domains:

1. First, define a partitioning of the time domain [0,T] into p time sub-intervals $[T_{j-1}, T_j], j = 1, ..., p, 0 = T_0 < T_1 < ... < T_p = T$;
2. For each $j = 1, ..., p$, solve the initial zero value problem
$$\dot{\boldsymbol{v}}_j(t) = -A\boldsymbol{v}_j(t) + \boldsymbol{f}(t), \quad \boldsymbol{v}_j(T_{j-1}) = 0, \quad t \in [T_{j-1}, T_j]; \tag{6}$$
3. For each $j = 1, ..., p$, solve the homogeneous problem
$$\dot{\boldsymbol{w}}_j(t) = -A\boldsymbol{w}_j(t), \quad \boldsymbol{w}_j(T_{j-1}) = \boldsymbol{v}_{j-1}(T_{j-1}), \quad t \in [T_{j-1}, T] \tag{7}$$
(with the notation $\boldsymbol{v}_0(T_0) := \boldsymbol{u}^0$).

It is clear that by a superposition principle, on can synthesize a solution $\boldsymbol{u}$ of (4) by the summation formula

$$\boldsymbol{u}(t) = \boldsymbol{v}_k(t) + \sum_{j=1}^{k} \boldsymbol{w}_j(t) \quad \text{for } k \text{ such that } t \in [T_{k-1}, T_k],\ k \in \{1, ..., p\}. \tag{8}$$

The PARAEXP algorithm is dedicated to parallel computing architectures, otherwise of course there is no benefit to execute it sequentially on one processor. It is remarkable to notice that good implementations of PARAEXP do not require any communication until the solution synthesis step, so the theoretical optimal efficiency is 1 before synthesis. Of course there is an issue of load balancing between processors because for a uniform time domain partitioning, some processors (especially the first one) are doing more work than others. The algorithm is graphically summarized in Fig. 1.

Another key of performance is the fast computation of the matrix exponentials. The solution of the homogeneous problem (7) in $[T_{j-1}, T]$ is

$$\boldsymbol{w}_j(t) = \exp(-At)\boldsymbol{v}_{j-1}(T_{j-1})$$

and thus has to be evaluated many times (at any time t in fact). There are many approaches to compute accurate approximate matrix exponentials as commented in [10]. A way is to

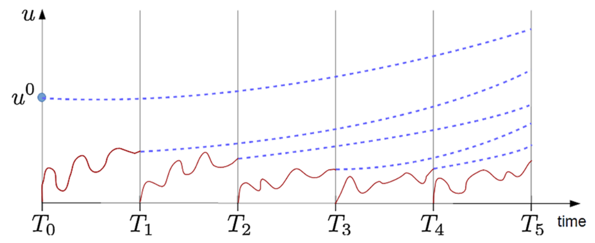

Fig. 1 Schematics of the PARAEXP algorithm [10]. *Solid red lines* represent the solutions of inhomogeneous problems with zero initial value on each time slice $[T_{j-1}, T_j]$, computed in parallel. *Blue dashed lines* represent the solutions of homogeneous problems on time intervals $[T_j, T]$, also computed in parallel

search approximations into the Krylov subspace K^M of dimension M:

$$K^M = \text{span}(\boldsymbol{u}^0, A\boldsymbol{u}^0, \dots, A^{M-1}\boldsymbol{u}^0),$$

looking for a best approximation into the truncated series expansion. We will come back on this issue when reduced order models (ROM) will be introduced below.

Nonlinear problems: an implicit-explicit IMEX time advance scheme

Hereafter, we switch to the nonlinear case considering the nonlinear heat equation as reference example. Before going into iterative and parallel algorithms, let us first consider a variant implicit-explicit (IMEX) time advance scheme introduced by Filbet, Negulescu and Yang [8] in 2012. The idea is to consider an implicit linear diffusion term with an upper bound of the thermal conductivity, and explicit remainding terms at the right hand side:

$$\frac{u^{n+1}-u^n}{\Delta t} - \underbrace{\nabla\cdot(\overline{\kappa}\nabla u^{n+1})}_{\text{linear, constant coefficients}} = \underbrace{-\nabla\cdot\left([\overline{\kappa}-\kappa(u^n)]\nabla u^n\right)}_{\text{varying coefficients, depend on } u} + f, \tag{9}$$

with

$$\overline{\kappa} \geq \sup_{x,t} \kappa(u(x,t)).$$

As demonstrated in [8], let us show that the semi-discrete in time scheme is stable in the one-dimensional case for a certain norm. Consider the homogeneous case $f = 0$ with homogeneous Neumann boundary conditions to simplify. We multiply (9) on u^{n+1} on the domain $\Omega = (0,1)$, hence we have

$$\begin{aligned}\frac{1}{2}\int_0^1 |u^{n+1}|^2\,ds - \frac{1}{2}\int_0^1 |u^n|^2\,ds &\leq \int_0^1 \left((u^{n+1})^2 - u^{n+1}u^n\right)\,ds \\ &\leq \Delta t\int_0^1 \left((\overline{\kappa}-\kappa(u^n))\partial_x u^n\partial_x u^{n+1} - \overline{\kappa}(\partial_x u^{n+1})^2\right)\,ds\end{aligned}$$

Let us recall the Peter-Paul inequality (extended Young's inequality): for any nonnegative real numbers a and b, we have $ab \leq \varepsilon a^2/2 + b^2(2\varepsilon)$ for every $\varepsilon > 0$. Using the assumption that $\kappa(u^n) \leq \overline{\kappa}$, and applying Peter-Paul's inequality with $a = |\partial_x u^n|$ we obtain

$$\begin{aligned}(\overline{\kappa}-\kappa(u^n))\partial_x u^n\partial_x u^{n+1} &\leq \frac{\varepsilon}{2}(\partial_x u^n)^2 + \frac{(\overline{\kappa}-\kappa(u^n))^2}{2\varepsilon}(\partial_x u^{n+1})^2 \\ &\leq \frac{\varepsilon}{2}(\partial_x u^n)^2 + \frac{\overline{\kappa}^2}{2\varepsilon}(\partial_x u^{n+1})^2.\end{aligned}$$

Therefore with the choice $\varepsilon = \overline{\kappa}$, we have the weighted Sobolev norm decrease

$$\frac{1}{2}\int_0^1 (u^{n+1})^2 dx + \frac{\overline{\kappa}}{2}\Delta t \int_0^1 |\partial_x u^{n+1}|^2 \, dx \le \frac{1}{2}\int_0^1 (u^n)^2 dx + \frac{\overline{\kappa}}{2}\Delta t \int_0^1 |\partial_x u^n|^2 \, dx.$$

This semi-discretization leads to a full discrete scheme in the form

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} + \overline{A}\, \boldsymbol{u}^{n+1} = \boldsymbol{g}(\boldsymbol{u}^n) \tag{10}$$

with $\boldsymbol{g}(\boldsymbol{u})$ in the form $\boldsymbol{g}(\boldsymbol{u}) = (\overline{A} - A(\boldsymbol{u}))\boldsymbol{u} + \boldsymbol{f}$. What is interesting of course is that the matrix of the implicit part is constant, and thus has to be assembled and factorized once. Moreover the system is linear in the variable $\boldsymbol{u}^{n+1}$. Unfortunately, the PARAEXP algorithm cannot be applied directly here because the right hand side $\boldsymbol{g}(\boldsymbol{u}^n)$ depends on the solution itself.

Iterative methods: the LATIN approach

A usual way to deal with nonlinear equations numerically is to use an iterative process within a fixed point algorithm. The LATIN (LArge Time INcremental) method pioneered by Ladevèze [15] and since then broadly used in computational structural Mechanics and material science (see [16] for a recent reference) solves time-dependent problems (linear or nonlinear) according to a two-step iterative process. To separate the difficulties, equations are partitioned into two groups: (i) a group of equations being local in space and time, possibly nonlinear (representing equilibrium equations for example); (ii) a group of linear equations, possibly global in the spatial variable. Then ad-hoc space-time approximations methods are used for the treatment of the global problem. Of course, space-time local equations can be solved in parallel, what makes the LATIN method efficient and suitable for today's HPC facilities. Let us emphasize that with LATIN, it is possible to solve hard nonlinear mechanics problems including thermodynamics irreversible problems (plasticity, friction as examples).

As an llustration, let us describe the LATIN method on the (rather simple) nonlinear heat problem:

1. Initialization ($k = 0$): let $u_{(0)} \in L^2((0,T), H^1(\Omega))$ an approximate solution (in space and time) of the nonlinear problem (it can be an approximate solution obtained with a coarse solver for example); compute $\tilde{\kappa}_{(0)} = \kappa(u_{(0)})$;
2. Iterate k, step 1 (global linear solution). Solve the linear problem

   $$\partial_t u_{(k+1)} - \nabla \cdot (\tilde{\kappa}_{(k)} \nabla u_{(k+1)}) = f$$

   with given initial and boundary conditions.
3. Iterate k, step 2 (local projection over the admissible manifold). Compute

   $$\tilde{\kappa}_{(k+1)} = \kappa(u_{(k+1)}).$$
4. Check convergence, $k \leftarrow k + 1$ if not and go to 2.

The step 1 performs a global (linear) evolution of the solution whereas a pointwise nonlinear projection on the equilibrium conductivity coefficients is done in step 2. We have a natural convergence indicator in terms of distance between the frozen conductivity $\tilde{\kappa}$ and $\kappa(u_{(k)})$:

$$e_{(k)} = \left\| \kappa(u_{(k+1)}) - \tilde{\kappa}_{(k)} \right\| .$$

In particular, if κ is Lipschitz continuous with Lipschitz constant L, then

$$e_{(k)} \le L \| u_{(k+1)} - u_{(k)} \|.$$

Remark that step 2 can be performed in parallel (in time).

For better and faster convergence, one can imagine variant approaches using a relaxation approach: remark first that $\kappa(u)$ is (formally) solution of the partial differential equation

$$\partial_t \kappa(u) - \nabla \cdot (\kappa(u) \nabla \kappa(u)) = \kappa'(u) f - \kappa''(u) \kappa(u) |\nabla u^2|.$$

If κ is a strictly convex function for example, then the second term at the right hand side is negative. One may consider the approximate (augmented) problem

$$\begin{cases} \partial_t u - \nabla \cdot (\tilde{\kappa} \nabla u) = f, \\ \partial_t \tilde{\kappa} - \nabla \cdot (\overline{\kappa}\, \nabla \tilde{\kappa}) = \kappa'(u) f + \dfrac{\kappa(u) - \tilde{\kappa}}{\varepsilon}. \end{cases} \tag{11}$$

where $\varepsilon > 0$ is a given relaxation time (assumed to be rather small). By this way, it is expected that $\tilde{\kappa}$ evolves much closer toward the value $\kappa(u)$. One can then derive an iterative process with again two steps (linear solution + projection) as in the LATIN method.

Newton and quasi-Newton approaches

For the sake of simplicity, let us consider here the initial value problem with general autonomous system of ordinary differential equations

$$\dot{\boldsymbol{u}} = \boldsymbol{f}(\boldsymbol{u}), \quad t \in (0, T], \tag{12}$$

with $\boldsymbol{f}$ assumed to be differentiable and Lipschitz continuous, and initial condition $\boldsymbol{u}(0) = \boldsymbol{u}^0$. The solution $\boldsymbol{u} \in L^2((0,T), \mathbb{R}^d)$ can be seen as the zero of a nonlinear operator $\boldsymbol{G}$,

$$\boldsymbol{G}(\boldsymbol{u}) := \dot{\boldsymbol{u}} - \boldsymbol{f}(\boldsymbol{u}) = 0.$$

The directional derivative of $\boldsymbol{G}$ at point $\boldsymbol{u}$ in the direction $\boldsymbol{v}$ is

$$D\boldsymbol{G}(\boldsymbol{u})\boldsymbol{v} = \dot{\boldsymbol{v}} - D\boldsymbol{f}(\boldsymbol{u})\, \boldsymbol{v}.$$

Then the standard Newton-Raphson method applied to $\boldsymbol{G}$ reads for the k-th iterate

$$D\boldsymbol{G}(\boldsymbol{u}_{(k)})(\boldsymbol{u}_{(k+1)} - \boldsymbol{u}_{(k)}) = -\boldsymbol{G}(\boldsymbol{u}_{(k)})$$

that simplifies into

$$\begin{aligned} \dot{\boldsymbol{u}}_{(k+1)} &= \boldsymbol{f}(\boldsymbol{u}_{(k)}) + D\boldsymbol{f}(\boldsymbol{u}_{(k)})(\boldsymbol{u}_{(k+1)} - \boldsymbol{u}_{(k)}) \\ &= D\boldsymbol{f}(\boldsymbol{u}_{(k)})\boldsymbol{u}_{(k+1)} + \big(\boldsymbol{f}(\boldsymbol{u}_{(k)}) - D\boldsymbol{f}(\boldsymbol{u}_{(k)})\boldsymbol{u}_{(k)}\big) \end{aligned} \tag{13}$$

Hence, the Newton-Raphson method provides a sequence of linear problems (of unknown $\boldsymbol{u}_{(k+1)}$) with variable coefficients and sources (depending on $\boldsymbol{f}$ and $\boldsymbol{u}_{(k)}$).

Spectral structure of the linearized problem

Let us emphasize that, at a given k, the linear system has the expected spectral structure for approximate solutions near an equilibrium $\overline{\boldsymbol{u}}$, that is $\boldsymbol{f}(\overline{\boldsymbol{u}}) = 0$. For $\boldsymbol{u}_{(k+1)}$ close to $\overline{\boldsymbol{u}}$, we have

$$\frac{d}{dt}(\boldsymbol{u}_{(k+1)} - \overline{\boldsymbol{u}}) = D\boldsymbol{f}(\boldsymbol{u}_{(k)})(\boldsymbol{u}_{(k+1)} - \overline{\boldsymbol{u}}) + \Big[\boldsymbol{f}(\boldsymbol{u}_{(k)}) - \boldsymbol{f}(\overline{\boldsymbol{u}}) - D\boldsymbol{f}(\boldsymbol{u}_{(k)})(\boldsymbol{u}_{(k)} - \overline{\boldsymbol{u}})\Big]$$

For $\boldsymbol{u}_{(k)}$ close to $\overline{\boldsymbol{u}}$ and $\boldsymbol{f} \in \mathscr{C}^2$ we have

$$\boldsymbol{f}(\boldsymbol{u}_{(k)}) - \boldsymbol{f}(\overline{\boldsymbol{u}}) - D\boldsymbol{f}(\boldsymbol{u}_{(k)})(\boldsymbol{u}_{(k)} - \overline{\boldsymbol{u}}) = O(|\boldsymbol{u}_{(k)} - \overline{\boldsymbol{u}}|^2),$$

then

$$\frac{d}{dt}(\boldsymbol{u}_{(k+1)} - \overline{\boldsymbol{u}}) \simeq D\boldsymbol{f}(\overline{\boldsymbol{u}})(\boldsymbol{u}_{(k+1)} - \overline{\boldsymbol{u}})$$

which is the expected linearized system.

Quasi-Newton approach

As an additional approximation, a quasi-Newton method will replace the Jacobian matrix $D\boldsymbol{f}(\boldsymbol{u}_k)$ by an approximate one $A_{(k)} \simeq D\boldsymbol{f}(\boldsymbol{u}_k)$, simpler to compute, thus giving the iterative process

$$\begin{aligned}\dot{\boldsymbol{u}}_{(k+1)} &= \boldsymbol{f}(\boldsymbol{u}_{(k)}) + A_{(k)}\,(\boldsymbol{u}_{(k+1)} - \boldsymbol{u}_{(k)}) \\ &= A_{(k)}\,\boldsymbol{u}_{(k+1)} + \big(\boldsymbol{f}(\boldsymbol{u}_{(k)}) - A_{(k)}\boldsymbol{u}_{(k)}\big)\end{aligned}$$

If we are able to build some coarse approximation $\boldsymbol{g}$ of $\boldsymbol{f}$ such that the quasi-Newton secant condition

$$A_{(k)}\,(\boldsymbol{u}_{(k+1)} - \boldsymbol{u}_{(k)}) = \boldsymbol{g}(\boldsymbol{u}_{(k+1)}) - \boldsymbol{g}(\boldsymbol{u}_{(k)}) \tag{14}$$

is satisfied, we get the Jacobian-free quasi-Newton iteration

$$\dot{\boldsymbol{u}}_{(k+1)} = \boldsymbol{f}(\boldsymbol{u}_{(k)}) + \big(\boldsymbol{g}(\boldsymbol{u}_{(k+1)}) - \boldsymbol{g}(\boldsymbol{u}_{(k)})\big) \tag{15}$$

or equivalently

$$\dot{\boldsymbol{u}}_{(k+1)} = \boldsymbol{g}(\boldsymbol{u}_{(k+1)}) + \big(\boldsymbol{f}(\boldsymbol{u}_{(k)}) - \boldsymbol{g}(\boldsymbol{u}_{(k)})\big)\,. \tag{16}$$

In (16), $\boldsymbol{g}(\boldsymbol{u}_{(k+1)})$ can be seen as a predictor term, whereas $(\boldsymbol{f}(\boldsymbol{u}_{(k)}) - \boldsymbol{g}(\boldsymbol{u}_{(k)}))$ is a corrector term toward $\boldsymbol{f}$ depending on the iterate (k) only. By construction we retrieve the accuracy of $\boldsymbol{f}$ at convergence. A quasi-Newton secant condition ensures superlinear convergence according to the Dennis and Moré theorem.

The PARAREAL method

The recent PARAREAL method, initially proposed by Lions et al. [17] in 2001, is nothing else but a parallel-in-time version of the quasi-Newton method (16) above. In PARAREAL, the time domain is decomposed into p subdomains. Then we define a double-index sequence of approximate solutions $\boldsymbol{u}^j_{(k)}$, where k still denotes the current index of the iterative process and j is the number of the time subdomain $[T_{j-1}, T_j]$. In its regular current form (see [3,4]), the PARAREAL algorithm is defined as follows:

1. Define a partition in time $[T_{j-1}, T_j]$, $0 = T_0 < T_1 < \ldots < T_p = T$;
2. Define a cheap coarse propagator $\mathscr{G}$ and a fine propagator $\mathscr{F}$.
3. Initialization ($k = 0$): $\boldsymbol{u}^0_{(0)} = \boldsymbol{u}^0$, $\boldsymbol{u}^{j+1}_{(0)} = \mathscr{G}(\boldsymbol{u}^j_{(0)})$;
4. Loop on the iterates k:
$$\boldsymbol{u}^{j+1}_{(k+1)} = \mathscr{G}(\boldsymbol{u}^j_{(k+1)}) + \left(\mathscr{F}(\boldsymbol{u}^j_{(k)}) - \mathscr{G}(\boldsymbol{u}^j_{(k)})\right) \tag{17}$$
5. Check convergence, test the stop criterion.

The PARAREAL algorithm is graphically represented in the schematics of Fig. 2. From (17) and the graph dependency of Fig. 2, one can understand that each corrector term on time slice j

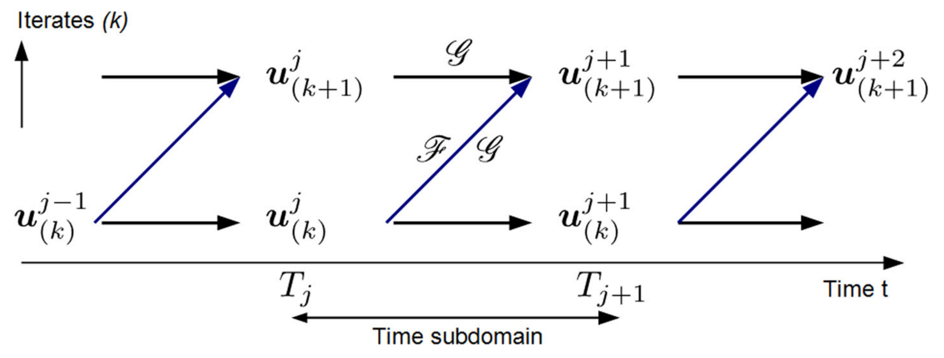
Iterates (k)
$\boldsymbol{u}^{j}_{(k+1)}$
$\mathscr{G}$
$\boldsymbol{u}^{j+1}_{(k+1)}$
$\boldsymbol{u}^{j+2}_{(k+1)}$
$\mathscr{F}/\mathscr{G}$
$\boldsymbol{u}^{j-1}_{(k)}$
$\boldsymbol{u}^{j}_{(k)}$
$\boldsymbol{u}^{j+1}_{(k)}$
T_j
T_{j+1}
Time t
Time subdomain

Fig. 2 Schematics and graph dependency into the PARAREAL algorithm

$$\left(\mathscr{F}(\boldsymbol{u}^{j}_{(k)}) - \mathscr{G}(\boldsymbol{u}^{j}_{(k)})\right)$$

can be evaluated in parallel over the p processors. On the other hand the coarse propagator term $\mathscr{G}(\boldsymbol{u}^{j}_{(k+1)})$ induces a persistent sequential part into the algorithm but it is expected to be evaluated quite fast. The trade-off is to design a fast, "accurate enough" coarse propagator which does not affect the whole performance of the algorithm.

One can imagine different choices of coarse solvers: low-order accurate time advances schemes, simplified equations, simplified models, discretizations on coarser meshes, etc. Reference papers like Bal and Maday [4] and Baffico et al. [3] show general convergence theorems for nonlinear ordinary differential systems using coarse time integrators as coarse solvers. Gander and Hairer in [9] also show a superlinear convergence of the parareal algorithm.

Putting all together

Actually, there are different ways to mix the strategies seen so far. As an example, let us still consider the nonlinear heat equation with time-varying source term:

$$\partial_t u - \nabla \cdot (\kappa(u)\nabla u) = f(t).$$

In the spirit of IMEX and LATIN, let us define the following iterative approach:

$$\partial_t u_{(k+1)} - \nabla \cdot (\overline{\kappa}\nabla u_{(k+1)}) = f(t) + \nabla \cdot ((\overline{\kappa} - \kappa_{(k)})\nabla u_{(k)}), \tag{18}$$

$$\kappa_{(k+1)} = \kappa(u_{(k+1)}). \tag{19}$$

On the left-hand side of the equation, we have replaced the thermal conductivity $\kappa(u)$ by some supremum as suggested by IMEX. In semi-discrete form, we get an equation in the form

$$\dot{\boldsymbol{u}}_{(k+1)} + \overline{A}\,\boldsymbol{u}_{(k+1)} = \boldsymbol{f}(t) + \boldsymbol{r}_{(k)}. \tag{20}$$

We get a linear equation for the unknown $\boldsymbol{u}_{(k+1)}$ with constant coefficient matrix $\overline{A}$, and the right hand side only depends on time through $f(t)$ and $\boldsymbol{u}_{(k)}(t)$. Then the PARAEXP algorithm can be applied at each iterative k. The remaining nonlinear operations like (19) and the assembling of $\boldsymbol{r}_{(k)}$ can be done in parallel (in time). In conclusion, we have replaced a nonlinear problem by a sequence of linear problems where some nonlinear evaluations have been sent into the right hand side, and so can be computed in parallel.

The Newton method to handle nonlinear terms with ROMs of dynamical systems

Reduced-order modeling is a general methodology to determine the principal information of a general high-dimensional problem and then reduce the problem, for example by projection. Reduction is generally possible when the M-Kolmogorov width

$$\delta_M(U) = \inf_{\substack{V^M \text{linear space, } V^M \subset V \\ \dim(V^M)=M}} \sup_{x\in U} \inf_{y^M\in V^M} \|x - y^M\|_V.$$

into an admissible close set U of a Banach space V is rather small for a rather small integer M (the dimension of the approximate space). One of the main motivations to do that is to strongly reduce the computational cost for the numerical solution. Even if there are recent advances in nonlinear reduced order modeling, in particular with the empirical interpolation method (EIM) proposed by Maday et al. [18], or discrete empirical interpolation method (DEIM) by Chaturantabut and Sorensen [5], there are still some issues and open problems for nonlinear time-dependent problems. Dealing with general nonlinear terms and reduced-order modeling for dynamical systems may be a difficult task, because:

- reduced-order models are expected to reproduce the stability of the system (for instance in the sense of Lyapunov, see [14] on this subject);
- the local dynamics has to be reproduced, at least "at first order", involving a compatibility of the spectral properties between full and reduced systems;
- the area visited by the trajectories into the state-space may be defined over a nonlinear manifold rather than in a linear subspace. Thus nonlinear dimensionality reduction methods would be better candidates for reduction.

Balanced truncation strategy [1,22] for example is a trade-off in the reduction process to provide sufficient accuracy for controllability and observability of dynamical systems. However the theory mainly deals with linear time-invariant (LTI) systems.

For time-dependent problems, one can adopt a greedy incremental strategy during time by adapting/enriching the low-dimensional subspace when the principal components are changing during time. But the price to pay is to online evaluate some (high-dimensional) nonlinear terms to control the error, what can be a penalizing factor of performance. If there is no other choice, parallel-in-time computing once again appears to be a complementary tool to keep global performance of the method.

Newton method and Galerkin projection method

Let us go back to the Newton method (13) that we rewrite here again

$$\dot{\boldsymbol{u}}_{(k+1)} = D\boldsymbol{f}(\boldsymbol{u}_{(k)})\boldsymbol{u}_{(k+1)} + \boldsymbol{f}(\boldsymbol{u}_{(k)}) - D\boldsymbol{f}(\boldsymbol{u}_{(k)})\boldsymbol{u}_{(k)}.$$

Let us consider a Galerkin approximation into the linear vector space

$$V^M_{(k)} = \text{span}(\boldsymbol{w}^1_{(k)}, \dots, \boldsymbol{w}^M_{(k)})$$

and assume that $(\boldsymbol{w}^\ell_{(k)}, \boldsymbol{w}^m_{(k)}) = \delta_{\ell m}$, $1 \le \ell, m \le M$. We are looking for an approximate rank-M solution $\boldsymbol{u}^M_{k+1}(t)$ in $V^M_{(k)}$, i.e.

$$\boldsymbol{u}^M_{(k+1)}(t) = \sum_{m=1}^{M} a^m_{(k+1)}(t)\, \boldsymbol{w}^m_{(k)} \tag{21}$$

for some real coefficients $a^m_{(k+1)}(t)$, $1 \le m \le M$ at time t. In order to get a reduced system, the Eq. (13) is projected onto the vector space $V^M_{(k)}$. Multipling (13) by any test function $\boldsymbol{v}^M \in V^M_{(k)}$, we look for a low-order solution $\boldsymbol{u}^M_{(k+1)}(t)$ in the form (21) such that

$$\left(\dot{\boldsymbol{u}}^M_{(k+1)}, \boldsymbol{v}^M\right) = \left(D\boldsymbol{f}(\boldsymbol{u}_{(k)})\boldsymbol{u}^M_{(k+1)}, \boldsymbol{v}^M\right) + \left(\boldsymbol{f}(\boldsymbol{u}_{(k)}) - D\boldsymbol{f}(\boldsymbol{u}_{(k)})\boldsymbol{u}_{(k)}, \boldsymbol{v}^M\right), \quad \forall \boldsymbol{v}^M \in V^M_{(k)}.$$

Taking $\boldsymbol{v}^M = \boldsymbol{w}^m_{(k)}$, $1 \le m \le M$, by orthogonality of the eigenvectors we get

$$\dot{a}^m_{(k+1)} = \sum_{\ell=1}^{M} a^\ell_{(k+1)}(D\boldsymbol{f}(\boldsymbol{u}_{(k)})\boldsymbol{w}^\ell_{(k)}, \boldsymbol{w}^m_{(k)}) + (\boldsymbol{f}(\boldsymbol{u}_{(k)}) - D\boldsymbol{f}(\boldsymbol{u}_{(k)})\boldsymbol{u}_{(k)}, \boldsymbol{w}^m_{(k)}).$$

In vector form, one obtains a reduced system in the form

$$\dot{\boldsymbol{a}}^M_{(k+1)} = \tilde{A}^M_{(k)}(t)\, \boldsymbol{a}^M_{(k+1)} + \boldsymbol{r}^M_{(k)}(t)$$

with $\boldsymbol{a}^M_{(k+1)}(t) = (a^m_{(k+1)}(t))_m$, $(\tilde{A}^M_{(k)})_{\ell m}(t) = (D\boldsymbol{f}(\boldsymbol{u}_{(k)}(t))\boldsymbol{w}^\ell_{(k)}, \boldsymbol{w}^m_{(k)})$ and $(\boldsymbol{r}^M_{(k)}(t))_m = (\boldsymbol{f}(\boldsymbol{u}_{(k)}(t)) - D\boldsymbol{f}(\boldsymbol{u}_{(k)}(t))\boldsymbol{u}_{(k)}, \boldsymbol{w}^m_{(k)})$. Remark that when the initial system is linear, i.e. $\boldsymbol{f}(\boldsymbol{u}) = A\boldsymbol{u}$, we retrieve the classical Galerkin projection over the space V^M:

$$\dot{\boldsymbol{a}}^M_{(k+1)} = \tilde{A}^M_{(k)}\, \boldsymbol{a}^M_{(k+1)}$$

with a constant matrix $\tilde{A}^M_{(k)}$, $(\tilde{A}^M_{(k)})_{\ell m} = (A\boldsymbol{w}^\ell_{(k)}, \boldsymbol{w}^m_{(k)})$. The assembling of both $\tilde{A}^M_{(k)}(t)$ and $\boldsymbol{r}^M_{(k)}(t)$ requires high-dimensional operations, but, fortunately, one can do this task in parallel (in time). Thus, one can nonetheless expect to get rather high performance. To summarize, at this stage of analysis, the algorithm of reduced-order modeling is the following:

1. (initialization). Use a coarse solver and compute $\boldsymbol{u}_{(0)}$. Loop over (k):
2. Compute M principal components $\boldsymbol{w}^m_{(k)}$, $m = 1, \ldots, M$ or a suitable reduced basis from the knowledge of $\boldsymbol{u}_{(k)}$.
3. Assemble and compute in parallel $\tilde{A}^M_{(k)}(t)$ and $\boldsymbol{r}^M_{(k)}(t)$ at all the discrete times.
4. Solve the linear problem

   $$\dot{\boldsymbol{a}}^M_{(k+1)} = \tilde{A}^M_{(k)}(t)\, \boldsymbol{a}^M_{(k+1)} + \boldsymbol{r}^M_{(k)}(t), \quad t \in (0, T]$$
   $$\boldsymbol{a}^M_{(k+1)}(0) = \boldsymbol{a}^0_{(k+1)} \in \mathbb{R}^M,$$

   and compute

   $$\boldsymbol{u}^M_{(k+1)}(t) = \sum_{m=1}^{M} a^m_{(k+1)}(t)\, \boldsymbol{w}^m_{(k)}.$$
5. Test convergence after iterate k.

Remark 1 For the computation of the basis functions $\boldsymbol{w}^m_{(k)}$, one can of course use Proper Orthogonal Decomposition (POD) [22] or any other dimensionality reduction method. The update the reduced basis may also be done by incrementing the basis set within an adaptive learning algorithm.

Remark 2 In the step 3, it is assumed that both $\tilde{A}_{(k)}(t)$ and $\boldsymbol{r}_{(k)}(t)$ have to be assembled and computed at all the discrete times. Of course, that may appear too penalizing for achieving high performance. Actually, one can consider additional reduction strategies for approximating both Jacobian matrix and right hand sides. This will be the aim of the following "Discussion" section.

Discussion about further reduction

There are many options to improve the whole numerical complexity of the algorithm using some additional approximations or reduction strategies.

Freezing up the Jacobian matrices

Let us go back to the Newton method

$$\dot{\boldsymbol{u}}_{(k+1)} = \boldsymbol{f}(\boldsymbol{u}_{(k)}) + D\boldsymbol{f}(\boldsymbol{u}_{(k)})(\boldsymbol{u}_{(k+1)} - \boldsymbol{u}_{(k)})$$

where the correction term $D\boldsymbol{f}(\boldsymbol{u}_{(k)})(\boldsymbol{u}_{(k+1)} - \boldsymbol{u}_{(k)})$ ensures quadratic convergence when it is converges. As already discussed in "Newton and quasi-Newton approaches", one can approximate the Jacobian matrix by some approximation $A_{(k)}(t)$ which is cheaper to evaluate, leading to the quasi-Newton approach

$$\dot{\boldsymbol{u}}_{(k+1)} = \boldsymbol{f}(\boldsymbol{u}_{(k)}) + A_{(k)}(t)\,(\boldsymbol{u}_{(k+1)} - \boldsymbol{u}_{(k)}).$$

The matrices $A_{(k)}$ still depend on time t a priori. But one could consider frozen approximates Jacobian matrices $A^j_{(k)}$ of time slices $[T_j, T_{j+1}]$, further inviting for a parallel-in-time strategy.

Adding coarse models

If we do not want to worry about Jacobian matrices, then the other option is to consider a coarse model $\boldsymbol{g}$ of $\boldsymbol{f}$ as mentioned in "Newton and quasi-Newton approaches" section. In this case, the quasi-Newton iteration reads

$$\dot{\boldsymbol{u}}_{(k+1)} = \boldsymbol{f}(\boldsymbol{u}_{(k)}) + \left(\boldsymbol{g}(\boldsymbol{u}_{(k+1)}) - \boldsymbol{g}(\boldsymbol{u}_{(k)})\right).$$

In order to achieve an efficient reduced-order model, one have now to deal with the nonlinear term $\boldsymbol{g}(\boldsymbol{u}_{(k+1)})$. An efficient and tractable way to proceed is to use an empirical interpolation method (EIM, [18]) for that. In that case, we can even make $\boldsymbol{g}$ depend on (k), according to some adaptive learning process (greedy algorithm, inflating basis, etc). Remark finally that the iterative process can once again be set up into a parallel-in-time framework following ideas from the PARAREAL algorithm.

Achieving dimensionality reduction for $\boldsymbol{f}$

If possible, one can also use a reduced-order approximation for $\boldsymbol{f}$. If the iterative algorithm is expected to converge towards a solution that has the same order of accuracy than the original one, one have to consider an accurate reduced-order model for $\boldsymbol{f}$. Once the empirical interpolation method may help us for that. However, if a global-in-time reduction strategy is considered, it is possible that the dimension M of the low-order vector space becomes too large, leading to a degradation of the whole performance.

An alternative approach would be to consider a family of local-in-time empirical interpolation methods for $\boldsymbol{f}$. In this case, we should also consider local models $\boldsymbol{f}^j_{(k)}$ available in the time slice $[T_j, T_{j+1}[$ which can also be updated at each k from a learning process.

Approximate exponential integrators

In order to make the PARAEXP algorithm globally efficient, it is essential to compute fast and accurate approximate exponential integrators. In the case of the linear heat equation, we have to compute the exponential of a large scale, symmetric sparse matrix A. More precisely, for the the problem $\dot{\boldsymbol{u}} = A\boldsymbol{u}$ with initial data $\boldsymbol{u}(0) = \boldsymbol{u}^0$, we have to compute the solution $\boldsymbol{u}(t) = \exp(tA)\boldsymbol{u}^0$ for any $t \in [0, T]$.

As mentioned in [9], there are numerous techniques to determine accurate exponential matrices. Among then, one can for example mention Padé approximants, exponentially fitted integration methods, or approximations based on projections over Krylov subspaces

$$K^M = [\boldsymbol{u}^0 \; A\boldsymbol{u}^0 \; A^2\boldsymbol{u}^0 \; \ldots \; (A^{M-1}\boldsymbol{u}^0)].$$

through Arnoldi orthogonalization iterations [12]. Actually the Krylov-Galerkin projection can be seen as a reduced-order technique, with a suitable reduced basis that fits action of matrix exponentials. But of course there are other choices of suitable basis functions like the first eigenvectors ϕ^m of A:

$$A\phi^m = \lambda^m \phi^m.$$

For A symmetric positive definite with eigenvalues arranged in increasing order, that is $0 < \lambda^1 \leq \lambda^2 \leq \ldots \lambda^M \leq \ldots$, it is natural to consider from the approximation error point of view the M first eigenvectors of A as vectors spanning the reduced approximation subspace. We will denote $\tilde{A}$ the projection of A on this discrete subspace and of course we have $\text{rank}(A) = M$. Considering the iterative approach of linear problems

$$\dot{\boldsymbol{u}}_{(k+1)} = \tilde{A}\boldsymbol{u}_{(k)} + (A - \tilde{A})\boldsymbol{u}_{(k)}, \quad t \in [0, T], \tag{22}$$

$$\boldsymbol{u}(0) = \boldsymbol{u}^0, \tag{23}$$

by superposition principle, one can first consider the low-order homogeneous problem

$$\dot{\boldsymbol{v}}_{(k+1)} = \tilde{A}\, \boldsymbol{v}_{(k+1)},$$
$$\boldsymbol{v}_{(k+1)}(0) = \boldsymbol{u}^0$$

for which we have an efficient low-order exponential solution, and on the other side the high-dimensional problem with zero initial value

$$\dot{\boldsymbol{w}}_{(k+1)} = \tilde{A}\, \boldsymbol{w}_{(k+1)} + (A - \tilde{A})\boldsymbol{u}_{(k)},$$
$$\boldsymbol{w}_{(k+1)}(0) = 0,$$

then $\boldsymbol{u}_{(k+1)}(t) = \boldsymbol{v}_{(k+1)}(t) + \boldsymbol{w}_{(k+1)}(t)$. In the spirit of the PARAEXP algorithm, one can set up the superposition principle within a parallel-in-time time decomposition to deal separately with low-order homogeneous exponential solution and high-dimensional inhomogeneous problems.

Closing discussion

From this review on efficient time-advance solvers including IMEX, LATIN, PARAEXP and PARAREAL algorithms, we try to show the different ways and tracks to deal with large-scale dynamical systems, linear and/or nonlinear terms. For the sake of an easy discussion, we have taken the example of the heat equation (linear or nonlinear). We are aware that this may be too restrictive and nonlinear computational mechanics including for example thermodynamics irreversible problems need more efforts and technical developments. Among the methods discussed above, some of them have been designed to address these problems. This is the case for the LATIN approach for example.

Time parallelization appears to be a promising key element of speedup. For problems with a small Kolmogorov width, reduced-order modeling may be a supplementary methodology to accelerate the whole time advance solution. For numerous reasons, it is interesting to cast a nonlinear problem into a sequence of linear problems within an iterative process. Linear problems are easier to deal with, and there are dedicated tools like the parallel-in-time PARAEXP method. On the other hand, an iterative process allows for achieving multi-fidelity adaptive solvers, using incremental, greedy or learning algorithms. Of course, we have to keep in mind that iterative methods may not converge. So in the design process of the numerical approach, one has to answer to the following questions: is the whole iterative process stable, is it possible to prove the convergence ? If the method is convergent, what is the rate of convergence ? Is it possible to accelerate the convergence ? At convergence, is it sure that the iterative algorithm converges to the solution obtaines with the accuracy we paid at the finest level ? For parallel algorithms, what is the effective speedup ?

Last but not least, managing multi-fidelity models and multi-level reduced-order models as well as parallel-in-time algorithms and learning algorithms implemented on distributed memory computer architecture necessarily require data management efforts and smart software engineering.

Conclusions

The first aim of this paper is to review different efficient time-advance solvers (including IMEX, PARAEXP, LATIN, PARAREAL) and show connections between them. We also try to show the links with quasi-Newton approaches and relaxation/projection methods to deal with nonlinear terms. Parallel-in-time algorithms appear to be a complementary and promising framework for the fast solution of time-dependent problems. Finally, reduced-order models (POD-based, principal eigenstructure, a priori reduced bases, ...) can be possibly included to achieve better performance. In a future paper, we will achieve numerical experiments on different hybrid approaches.

Received: 13 November 2015 Accepted: 2 March 2016
Published online: 15 March 2016

References

1. Antoulas AC. An overview of approximation methods for large-scale dynamical systems. Annu Rev Control. 2005;29:181–90.
2. Audouze C, De Vuyst F, Nair PB. Nonintrusive reduced-order modeling of parametrized time-dependent partial differential equations. Numeri Methods Partial Differen Equ. 2013;29(5):1587–628.
3. Baffico L, Bernard S, Maday Y, Turinici G, Zrah G. Parallel-in-time molecular dynamics simulations. Phys Rev E. 2002;66(5):057701.
4. Bal G, Maday Y. A "parareal" time discretization for nonlinear PDEs with application to the pricing of an American put. Recent developments in domain decomposition methods. Berlin: Springer; 2002. p. 189–202.
5. Chaturantabut S, Sorensen DC. Nonlinear model reduction via discrete empirical interpolation. SIAM J Sci Comput. 2010;32(5):2737–64.
6. Chinesta F, Ammar A, Lemarchand F, Beauchène P, Boust F. Parallel time integration and high resolution homogenization. CMAME. 2008;197(5):400–13.
7. Cortial J, Fahrat C, Guibas LJ, Rajashekhar M. Compressed sensing and time-parallel reduced-order modeling of structural health monitoring using DDDAS. Computational science-ICCS 2007. Berlin: Springer; 2007. p. 1171–9.
8. Filbet F, Negulescu C, Yang C. Numerical study of a nonlinear heat equation for plasma Physics. Int J Comp Math. 2012;89(8):1060–82.
9. Gander MJ, Hairer E. Nonlinear convergence analysis for the PARAREAL algorithm. In: Domain decomposition methods in Science and Engineering. 2008. vol. 60, p. 4556.
10. Gander MJ, Güttel S. PARAEXP: a parallel integrator for linear initial-value problems. SIAM J Sci Comput. 2013;35(2):C123–42.
11. Gander MJ. 50 years of time parallel time integration. Multiple shooting and time domain decomposition. Berlin: Springer; 2015.
12. Hochbruck M, Lubich C. On Krylov subspace approximations to the matrix exponential operator. SIAM J Num Anal. 1997;34(5):1911–25.

13. Kalashnikova I, Barone MF. On the stability of a Galerkin reduced order model (ROM) of compressible flow with solid wall and far-field boundary treatment. IJNME. 2010;83:1345–75.
14. Kalashnikova I, Barone MF, Arunajatesan S, von Bloemen Waanders BG. Construction of energy-stable projection-based reduced order models. Appl Math Comp. 2014;249:569–96.
15. Ladevèze P. Non linear computational structural mechanics new approaches and non-incremental methods of calculation. New York: Springer-Verlag; 1999.
16. Ladevèze P, Passieux JC, Néron D. The LATIN multiscale computational method and the proper generalized decomposition. Comput Methods Appl Mech Eng. 2010;199(21):1287–96.
17. Lions J, Maday Y, Turinici G. A "parareal" in time discretization of PDE's. Comptes Rendus de l'Académie des Sciences, Séries I, Mathematics. 2001;332(7):661–8.
18. Maday Y, Nguyen NC, Patera AT, Pau GSH. A general multipurpose interpolation procedure: the magic points. Commun Pure Appl Anal. 2009;81:383–404.
19. Minion M. A hybrid parareal spectral deferred corrections method. Commun Appl Math Comput Sci. 2010;5(2):265–301.
20. Nievergelt J. Parallel methods for integrating ordinary differential equations. Comm ACM. 1964;7:731–3.
21. Prud'homme C, Rovas D, Veroy K, Maday Y, Patera AT, Turinici G. Reliable real time solution of parametrized partial differential equations: reduced-basis output bound methods. J Fluids Eng. 2002;124(1):70–80.
22. Willcox K, Peraire J. Balanced model reduction via the proper orthogonal decomposition. AIAA J. 2002;40(11):2323–30.