Coupling Parareal and Waveform Relaxation Methods for Power Systems

Thomas Cadeau, Frédéric Magoulès
Applied Mathematics and Systems Laboratory
Ecole Centrale Paris
Châtenay-Malabry, France
Email: frederic.magoules@hotmail.com

***Abstract*—In this paper, a coupling strategy of the Parareal method and the Waveform Relaxation method is presented for power systems analysis. After a brief presentation of the classical Waveform Relaxation method and the Parareal method, the coupled Parareal-Waveform Relaxation method, recently introduced for the solution of the heat equation, is introduced. This coupled method is here extended to the solution of large power systems. Applications on a simplified European electricity network illustrate the impressive performance of this new method for the solution of large power systems on parallel architectures.**

***Index Terms*—Domain decomposition method; parareal method; waveform relaxation method; parallel and distributed computing; differential algebraic equations; power systems**

I. Introduction

With the complexity of electricity network, power system dynamic simulation is becoming more and more challenging. Interconnected networks of countries, for instance, lead to the simulation of large power systems involving hundreds of thousands of variables. For such systems, new numerical methods must be designed to achieve fast and near real-time computation.

In this paper, after a brief presentation of the mathematical equations arising from power systems, the Waveform Relaxation method [13] and the Parareal method [5] are presented. Then a coupled Parareal-Waveform Relaxation method, originally introduced in [2] for the heat equation is described. This new method leads by construction to a higher speedup compared to the Waveform Relaxation method and the Parareal method. In this paper, this coupled Parareal-Waveform Relaxation method is extended to the equations arising from power systems and numerical experiments performed on a simplified European electricity network illustrate the extraordinary performance of this new method.

II. Power Systems Equations

The mathematical system [11] to solve is a system of the form:

$$\left\{ \begin{array}{lcl} y' & = & f(y,z,t),\ t\ \in [0,T] \\ 0 & = & g(y,z,t), \end{array} \right. \tag{1}$$

where f and g are functions of the differential variables, y, the algebraic variables, z, and the time variable, t. Function f corresponds mainly to the rotor electrical and mechanical equations of all the machines. It is assumed in these equations that each machine is strictly coupled to other machines through the network. Equation $0 = g(y,z,t)$ corresponds to the stator equations of each machine and to the equations of the network and loads. Due to the properties of the power system (events, faults, etc.), the functions f and g are nonlinear functions, nondifferentiable, neither continuous. Anyway, the functions f and g are smooth enough between two discontinuities, which allows to consider linear approximation of f and g in such interval. Fixing the functions f_0 and g_0 at $t = 0$, the system to solve becomes the following:

$$\left\{ \begin{array}{lcl} y' & = & M\delta y + N\delta z + f_0,\ t\ \in [0,T] \\ 0 & = & P\delta y + Q\delta z + g_0, \end{array} \right. \tag{2}$$

with $\delta y = y - y_0$, $\delta z = z - z_0$, where y, z are the states at time $t \in [0,T]$, and y_0, z_0 are the states at time $t = 0$. The integration scheme used for the differential equations is a backward differentiation formula, using a trapezoidal rule. The implicit system to solve can be written as:

$$A \begin{pmatrix} \delta y_{n+1} \\ \delta z_{n+1} \end{pmatrix} = B \begin{pmatrix} \delta y_n \\ \delta z_n \end{pmatrix} + b, \tag{3}$$

where:

$$A = \begin{pmatrix} I - \frac{h}{2}M & -\frac{h}{2}N \\ P & Q \end{pmatrix},$$

$$B = \begin{pmatrix} I + \frac{h}{2}M & \frac{h}{2}N \\ 0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} hf_0 \\ -g_0 \end{pmatrix}.$$

Solving the system (3) implies a matrix-vector multiplication by the matrix B and the solution of a linear system with the matrix A. The matrices A and B, and the vector b depend only of the timestep h, which is here fixed for the simulation. Once the right-hand side (rhs) of (3) is evaluated, a direct method is used to solve the linear system with the matrix A. The factorization of the matrix A is done once in the simulation and only a forward/back substitution is performed at each timestep. Finally, the solution is rebuilt from the initial values:

$$\begin{pmatrix} y_{n+1} \\ z_{n+1} \end{pmatrix} = \begin{pmatrix} \delta y_{n+1} \\ \delta z_{n+1} \end{pmatrix} + \begin{pmatrix} y_0 \\ z_0 \end{pmatrix}. \tag{4}$$

III. Related Works

A. Domain decomposition methods

Domain decomposition methods [9], [12], [8] allows to solve large problems in parallel through artificial subdivisions of the domain into smaller subproblems. Domain decomposition strategies involve: a decomposer to split a domain into subdomains; local solvers to find solutions for the subproblems [9]; interface conditions [7], [1] enforcing compatibility and equilibrium between the subdomains, which might be tuned to ensure fast convergence of the agorithm [6]; and a solution strategy for the interface problem. To ensure good performances of domain decomposition methods, an appropriate partitioning of the domain is mandatory i.e. that the number of variables are well balanced between the subdomains, and the size of the interfaces between the subdomains remains as small as possible.

Given a subdomain named (s), the indexes of the variables belonging to the internal area of the subdomain or to the overlapping area of the neighboring subdomain are named (i_s). The indexes of the variables belonging to the neighboring subdomain in contact with the indexes of the variables (i_s) are named (b_s). Each subsystem s depends on the variables $(\delta_y^{(b_s)}, \delta_z^{(b_s)})^t$ which are computed by the neighboring subsystem. The global system (2) is projected on the partitioning of the domain and Dirichlet conditions are defined on the interface between the subdomains. With these notations, each local subsystem can be written in the form:

$$\begin{pmatrix} y^{(i_s)\prime} \\ 0 \end{pmatrix} = J^{(i_s)} \begin{pmatrix} \delta y^{(i_s)} \\ \delta y^{(b_s)} \\ \delta z^{(i_s)} \\ \delta z^{(b_s)} \end{pmatrix} + \begin{pmatrix} f_0^{(i_s)} \\ g_0^{(i_s)} \end{pmatrix}, \quad (5)$$

where $J^{(i_s)} = \begin{pmatrix} M^{(i_s)} & M^{(b_s)} & N^{(i_s)} & N^{(b_s)} \\ P^{(i_s)} & P^{(b_s)} & Q^{(i_s)} & Q^{(b_s)} \end{pmatrix}$ and after linearization, each subsystem can be written as follows:

$$A^{(i_s)} \begin{pmatrix} \delta y_{n+1}^{(i_s)} \\ \delta z_{n+1}^{(i_s)} \end{pmatrix} = B^{(i_s)} \begin{pmatrix} \delta y_n^{(i_s)} \\ \delta z_n^{(i_s)} \end{pmatrix} + b^{(i_s)}, \quad (6)$$

where $A^{(i_s)}$, $B^{(i_s)}$, $b^{(i_s)}$ are defined by:

$$A^{(i_s)} = \begin{pmatrix} I^{(i_s)} - \frac{h}{2}M^{(i_s)} & -\frac{h}{2}N^{(i_s)} \\ P^{(i_s)} & Q^{(i_s)} \end{pmatrix},$$

$$B^{(i_s)} = \begin{pmatrix} I^{(i_s)} + \frac{h}{2}M^{(i_s)} & \frac{h}{2}N^{(i_s)} \\ 0 & 0 \end{pmatrix},$$

$$b^{(i_s)} = \begin{pmatrix} hf_0^{(i_s)} \\ -g_0^{(i_s)} \end{pmatrix} + \begin{pmatrix} hM^{(b_s)} & hN^{(b_s)} \\ -P^{(b_s)} & -Q^{(b_s)} \end{pmatrix} \begin{pmatrix} \delta y_{n+1}^{(b_s)} \\ \delta z_{n+1}^{(b_s)} \end{pmatrix}.$$

B. Waveform Relaxation method

To solve these subproblems, the Waveform Relaxation [13] method introduces an iterative scheme, with an iteration number k, as:

$$A^{(i_s)} \begin{pmatrix} \delta y_{n+1}^{(i_s),k+1} \\ \delta z_{n+1}^{(i_s),k+1} \end{pmatrix} = B^{(i_s)} \begin{pmatrix} \delta y_n^{(i_s),k} \\ \delta z_n^{(i_s),k} \end{pmatrix} + b^{(i_s),k}, \quad (7)$$

where

$$b^{(i_s),k} = \begin{pmatrix} hM^{(b_s)} & hN^{(b_s)} \\ -P^{(b_s)} & -Q^{(b_s)} \end{pmatrix} \begin{pmatrix} \delta y_{n+1}^{(b_s),k} \\ \delta z_{n+1}^{(b_s),k} \end{pmatrix} + \begin{pmatrix} hf_0^{(i_s)} \\ -g_0^{(i_s)} \end{pmatrix}.$$

Each local subproblem is allocated to a different processor and is solved in parallel. Communications between processes occur at each iteration since process s needs to receive $\delta y_{n+1}^{(b_s),k}$ and $\delta z_{n+1}^{(b_s),k}$. The time interval $[0, T]$ is split into W windows $[T_w, T_{w+1}]$ such as

$$0 = T_0 < T_1 < ... < T_w < T_{w+1} < ... < T_W = T,$$

and the window size is fixed to S timesteps. As a consequence, time T_w is equal to $t_{S.w} = w.S.h$ and $(\delta y_{S.w}^{(i_s)}, \delta z_{S.w}^{(i_s)})^t$ is the solution at time T_w. Knowing $\delta y_n^{(b_s),k}$ and $\delta z_n^{(b_s),k}$ for $n \in \{w.S, ..., (w+1).S\}$, the process s computes $\delta y_n^{(i_s),k+1}$ and $\delta z_n^{(i_s),k+1}$ for $n \in \{w.S, ..., (w+1).S\}$. To initialize the iterations, initial boundary values need to be defined. Using a constant fixed to the last known value at time $w.S$ allows to have a cheap and coherent initial guess:

$$\begin{pmatrix} \delta y_n^{(i_s),0} \\ \delta z_n^{(i_s),0} \end{pmatrix} = \begin{pmatrix} \delta y_{w.S}^{(i_s)} \\ \delta z_{w.S}^{(i_s)} \end{pmatrix}, \ n \in \{w.S, ..., (w+1).S\}. \quad (8)$$

At the convergence, the solution is finally rebuilt as:

$$\begin{pmatrix} y_{n+1}^{(i_s)} \\ z_{n+1}^{(i_s)} \end{pmatrix} = \begin{pmatrix} \delta y_{n+1}^{(i_s)} \\ \delta z_{n+1}^{(i_s)} \end{pmatrix} + \begin{pmatrix} y_0^{(i_s)} \\ z_0^{(i_s)} \end{pmatrix}. \quad (9)$$

C. Parareal method

Opposite to the waveform relaxation method, the Parareal method [5], [10], [3] aims to split the time domain and to solve each associated subproblem independently. The time domain $[0, T]$ is split into time windows $[T_w, T_{w+1}]$ such as

$$0 = T_0 < T_1 < ... < T_w < T_{w+1} < ... < T_W = T.$$

The objective is to solve the initial problem on each time window, in parallel. To ensure good load balancing, each time window contains the same number S of timestep h, i.e. $T_{w+1} - T_w = \Delta T = S.h$. Obviously, to solve the problem on $[T_w, T_{w+1}] = [t_{S.w}, t_{S.(w+1)}]$, the initial solution at time $T_w = t_{S.w}$ should be known.

Let's define $F_{\Delta T}$ a propagator which allows to compute the solution on the time window $[T_w, T_{w+1}]$, knowing the initial solution at time T_w. This propagator is based on the sequential solver, and is called the Fine propagator in the following. Let's define $C_{\Delta T}$ a cheap propagator which allows to compute an approximation of the solution at time T_{w+1} knowing an approximation of the solution at time T_w. This propagator is based on the sequential solver, but with $h = \Delta T$, and is called the Coarse propagator in the following. The matrix associated to $F_{\Delta T}$ (resp. $C_{\Delta T}$) is noted A_F (resp. A_C). The coarse propagator is computed sequentially, this allows to have an approximation of all $(\delta y_{S.w}, \delta z_{S.w})^t$. These approximate solutions allow the fine solver to compute the solution on each time window in parallel. Then an additional computation of the coarse solver, using the values computed by the fine solver, enables to introduce the iteration scheme, with an iteration number k, of the Parareal method:

$$\left\{ \begin{array}{lcl} \begin{pmatrix} \delta y_{n+1}^0 \\ \delta z_{n+1}^0 \end{pmatrix} & = & C_n^0 \\ \begin{pmatrix} \delta y_{n+1}^{k+1} \\ \delta z_{n+1}^{k+1} \end{pmatrix} & = & C_n^{k+1} + F_n^k - C_n^k \end{array} \right. \tag{10}$$

where

$$C_n^{k+1} = C_{\Delta T}(\begin{pmatrix} \delta y^{k+1} \\ \delta z_n^{k+1} \end{pmatrix}), F_n^{k+1} = F_{\Delta T}(\begin{pmatrix} \delta y_n^k \\ \delta z_n^k \end{pmatrix}).$$

IV. Coupling Parareal and Waveform Relaxation Methods

The Waveform Relaxation method and the Parareal method have a limited speedup. For the Waveform Relaxation method, the communications between the subdomains cost should be small compared to the solution cost of the local solution. It is thus important to bound the number of subdomains, i.e. the number of processes, in such a way that the number of variables within each subdomain is large enough. The Parareal method has a speedup bounded by the number of processes divided by two. Coupling the Waveform Relaxation method and the Parareal method allows to combine the good speedup of the Waveform Relaxation, with the maximum possible number of subdomains, i.e., before its speedup decreases, and to continue to increase its speedup by using the Parareal method.

In this study, the coupled Parareal-Waveform Relaxation method is based on the Parareal method, where (i) the Fine propagator $F_{\Delta T}$ is based on the Waveform Relaxation method (with the same time windows than the Parareal method $[T_w, T_{w+1}]$, and with the same timestep h than the sequential method); and where (ii) the Coarse propagator $C_{\Delta T}$ is based on the Waveform Relaxation method (with both the timestep and the window size equal to the Parareal time windows: $h = T_{w+1} - T_w$). To optimize the method, the number of iterations of the Fine and the Coarse propagators is bounded to a small number, 2 or 3 maximum. These new propagators with limited number of iterations are denoted $\tilde{F}_{\Delta T}$ and $\tilde{C}_{\Delta T}$. The opposite algorithm shows a parallel implementation of the coupled Parareal-Waveform Relaxation method. Each group of processors used to compute $\tilde{F}_{S.w}^k$ and $\tilde{C}_{S.w}^k$ by Waveform Relaxation is called a communicator.

V. Theoretical Speedups

Let T_h be the CPU time to compute one timestep using the Forward/Back substitutions for the global system, and N the number of computed timesteps for the considered simulation. The total CPU time of the sequential method is then equal to $T_{seq} = NT_h$.

For this theoretical analysis, the computational cost of the Forward/Back substitutions of A (or $A^{(i_s)}$) is assumed to evolve linearly with the subsystem size, i.e. that the CPU time to compute one timestep on a partition of P domains is equal to $T_P = T_h/P$. So the CPU time of the Waveform Relaxation

```
Factorization of $A_C$ and $A_F$
Recv $\begin{pmatrix} \delta y_{S.w}^0 \\ \delta z_{S.w}^0 \end{pmatrix}$ form previous communicator
Compute $\tilde{C}_{S.w}^0$ by Waveform Relaxation
Send $\begin{pmatrix} \delta y_{S.(w+1)}^0 \\ \delta z_{S.(w+1)}^0 \end{pmatrix}$ to following communicator
while Parareal has not converged, iterate on $k$ do
    Compute $\tilde{F}_{S.w}^k$ by Waveform Relaxation
    Recv $\begin{pmatrix} \delta y_{S.w}^{k+1} \\ \delta z_{S.w}^{k+1} \end{pmatrix}$ form previous communicator
    Compute $\tilde{C}_{S.w}^{k+1}$ by Waveform Relaxation
    Update $\begin{pmatrix} \delta y_{S.(w+1)}^{k+1} \\ \delta z_{S.(w+1)}^{k+1} \end{pmatrix}$ with (10)
    Send $\begin{pmatrix} \delta y_{S.(w+1)}^k \\ \delta z_{S.(w+1)}^k \end{pmatrix}$ to following communicator
end
Update the solution
```

method, with a convergence in K iterations on each of the W windows using P processors is equal to:

$$T_{WR} = WK\frac{N}{W}T_P = \frac{KN}{P}T_h.$$

The propagators of the Parareal method are based on the sequential method. Thus the CPU time to compute one timestep using the Fine (resp. Coarse) propagator, is $T_F = T_h$ (resp. $T_C = T_h$). The CPU time of the Parareal method, with a convergence in K iterations on W windows (and W processors) is:

$$T_{Pa} = \sum_{i=0}^{K-1} (W - i) T_C + (K-1)\frac{N}{W}T_F,$$

$$T_{Pa} = KWT_C - \frac{(K-1)K}{2}T_C + (K-1)\frac{N}{W}T_F.$$

The CPU times of the Parareal and the Waveform Relaxation methods allow to write the theoretical CPU time of the coupled method for a number of space domains equal to P, a number of windows equal to W, and a number of processors equal to WP. Let T_{FP} (resp. T_{CP}) denote the CPU time of one timestep of the Waveform Relaxation Fine (resp. Coarse) propagator on P subdomains. Let k_F and k_C denote the number of iterations bound of the Fine and the Coarse propagator. The CPU time of the coupled method is then:

$$T_{PWR} = Kk_C WT_{CP} + (K-1)k_F\frac{N}{W}T_{FP},$$

$$T_{PWR} = K\frac{k_C W}{P}T_C + (K-1)\frac{k_F N}{WP}T_F.$$

This leads to the following theoretical speedups for the Waveform Relaxation, Parareal and coupled Parareal-Waveform relaxation methods:

$$S_{WR} = \frac{P}{K},$$

$$S_{Pa} = \frac{1}{\frac{KW}{N} + \frac{(K-1)}{W} - \frac{(K-1)K}{2N}},$$

$$S_{PWR} = \frac{1}{\frac{Kk_C W}{PN} + \frac{(K-1)K}{WP}}.$$

VI. Numerical Results

The simulation of a simplified European electricity network is performed. The system consists of a total of 100.000 differential and algebraic equations. The system is linearized after an event (a load variation), which allows to compute several seconds of simulations and to obtain coherent values. To solve this problem, the system is split into 2, 4, 8, 16, 32 subsystems (in space) with the METIS library [4].

The Waveform Relaxation method is first considered. Each subsystem is allocated to one different processor. As illustrated in Fig. 1, the speedup of the Waveform Relaxation method increases till 16 subsystems. After reaching a peak, to have more subsystems does not increase the speedup, but on the contrary decreases it. This is coherent, since when more than 16 subsystems are considered, each subsystem contains not enough variables to ensure a good efficiency of the method.

Second, the Parareal method is considered and results are reported in Fig. 1.

In order to improve the speedup, the coupled Parareal-Waveform Relaxation method is now applied. The system (in space) is split into 16 subsystems (in space) and into till 32 windows (in time), forming a space-time grid. Each window contains 10 timesteps. The number of iterations of the Fine and the Coarse propagator k_F and k_C is fixed to 2. Each subsystem of the space-time grid, is handled by one different processor. The coupled Parareal-Waveform Relaxation method illustrated in Fig. 1 shows a continuing increasing speedup when considering more processes.

The results reported in Fig. 2 illustrate the respective speedups of the methods upon the number of subdomains. The number of processors is the number of space domains P for the Waveform Relaxation method, the number of time domains W for the Parareal method and PW for the coupled method, with $W = 32$ time domains and P the number of space domains. The corresponding theoretical speedup is also plotted in dot, using the number of iterations given by the numerical simulations. As we can see on this Figure, when applied to power systems, the coupled Parareal-Waveform Relaxation method clearly outperforms (both in theory and in practice) the Waveform Relaxation method and the Parareal method.

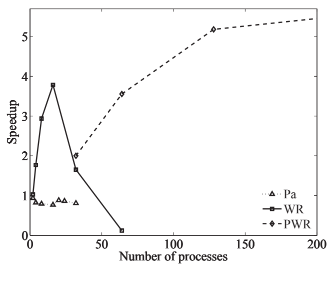

Fig. 1. Comparison of the speedup of the Waveform Relaxation method, the Parareal method and the coupled Parareal-Waveform Relaxation method.

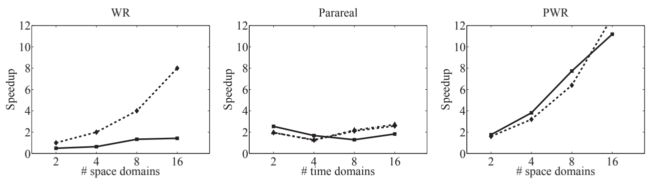

Fig. 2. Comparison of theoretical and numerical speedups of the Waveform Relaxation method, the Parareal method and the coupled Parareal-Waveform Relaxation method.

VII. Conclusions

In this paper a new coupled Parareal-Waveform Relaxation method has been presented for power systems. This method is based on the classical Waveform Relaxation method and the Parareal method. The numerical results obtained on a large simplified European electricity network illustrates the great performances of this new coupled Parareal-Waveform Relaxation method, over other methods when using large number of processors.

References

[1] M. Gander, L. Halpern, and F. Magoulès. An optimized Schwarz method with two-sided Robin transmission conditions for the Helmholtz equation. *International Journal for Numerical Methods in Fluids*, 55(2):163–175, 2007.

[2] R. Gueta and Y. Maday. Coupling the Parareal algorithm with overlapping and nonoverlapping domain decomposition methods. Lecture Notes in Computational Science and Engineering. Springer, (in press).

[3] D. Guibert and D. Tromeur-Dervout. Adaptive parareal for systems of ODEs. In O. B. Widlund and D. E. Keyes, editors, *Proceedings of the International Domain Decomposition Conference*, volume 55 of *Lecture Notes in Computational Science and Engineering*, pages 587–594. Springer Berlin Heidelberg, 2007.

[4] G. Karypis and V. Kumar. *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices Version 4.0*, 1998.

[5] J.-L. Lions, Y. Maday, and G. Turinici. Résolution d'EDP par un schéma en temps pararéel. *Comptes Rendus de l'Académie des Sciences - Série I - Mathematics*, 332(7):661–668, 2001.

[6] Y. Maday and F. Magoulès. Optimal convergence properties of the FETI domain decomposition method. *International Journal for Numerical Methods in Fluids*, 55(1):1–14, 2007.

[7] Y. Maday and F. Magoulès. Optimized Schwarz methods without overlap for highly heterogeneous media. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1541–1553, 2007.

[8] F. Magoulès and F.-X. Roux. Lagrangian formulation of domain decomposition methods: a unified theory. *Applied Mathematical Modelling*, 30(7):593–615, 2006.

[9] B. Smith, P. Bjorstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, UK, 1996.

[10] G. A. Staff and E. M. Rønquist. Stability of the parareal algorithm. In *Proceedings of the 15th International Domain Decomposition Conference*, Lecture Notes in Computational Science and Engineering, pages 449–456. Springer, 2003.

[11] B. Stott. Power system dynamic response calculations. In *Proceedings of the IEEE*, volume 67, pages 219–241, 1979.

[12] A. Toselli and O. Widlund. *Domain Decomposition methods: Algorithms and Theory*. Springer, 2005.

[13] J. White and A. Sangiovanni-Vincentelli. *Relaxation Techniques for the Simulation of VLSI Circuits*. Kluwer Academic Publishers, 1987.