VII International Conference on Computational Plasticity
COMPLAS 2003
E. Oñate and D.R.J. Owen (Eds)

A MULTIGRID METHOD FOR THE THERMOMECHANICAL BEHAVIOUR SIMULATION APPLICATED TO POWER ELECTRONICS CONVERTERS

L. Menanteau*, S. Caperaa†, and O. Pantalé†

*Power Electronics Associated Research Laboratory
Alstom Transport Tarbes
Rue du Docteur Guinier - BP4 - 65600 Semeac, France
e-mail: laurent.menanteau@lab-pearl.com, web page: http://lab-pearl.com

†Laboratoire Génie de Production (LGP)
Ecole Nationale d'Ingénieurs de Tarbes (ENIT)
47, Avenue d'Azereix - BP 1629 - 65016 Tarbes Cedex
Email: caperaa@enit.fr, pantale@enit.fr, Web page: http://www.enit.fr

Key words: Multigrid, Iterative Method, Software Development, Object Oriented Design

Abstract.
Power converters are an example of complex systems supporting physical phenomena located on very different-sized areas and for very different time scales. One of the goals of the PEARL Laboratory of Alstom Transport is to provide more efficient tools for the thermomechanical simulation (including plastic deformations and residual stresses) of such structures. To keep the possibility of modelling the physics of these areas (and not to use equivalent models such as springs, thermal resistances...), we need to build very large-scale problems, for which iterative solvers are more suitable.
This paper presents an implementation in C++ of a multigrid method realised on a home made software object-oriented finite element program. Different iterative solvers (Jacobi, Gauss-Seidel) and preconditionners (diagonal, SSOR) are used and different iterating processes between coarse(s) and fine grids are compared (V,W). The method is validated on a simplified case, showing the advantages in this particular field of thermomechanical simulation in terms of precision and future parallelization (domain decomposition methods with incompatible grids).

1 INTRODUCTION

The Alstom-PEARL laboratory, based in the Alstom Transport manufactury of Tarbes, designs power electronics converters used in railway transport. The goal of this research laboratory is to design new generations of power converters using new technologies. The main problems encountered in thermo-mechanical simulation of power converters are the different space and time scales, leading to large sized problems.

Figure 1: A power electronics converter

One way of research is to build models specific to these large scale (multidomains, multiphysics, multitimes) problems. Therefore, we are interested in multigrid methods. In this paper, we first make a short state of the art in the multigrid domain, then we present the developments based on a home-made object oriented software; finally, two basic test cases are validated.

2 MULTIGRID SOLVERS BASIC PRINCIPLES

2.1 Iterative solvers

Most of Finite Element solvers use iterative methods. The best interest of these numerical methods is to solve large scale problems more efficiently than direct methods: instead of inverting the system related to the discretized problem (on a given domain of study), the solution vector is built step by step following an iterative scheme (Jacobi, Gauss-Seidel, Conjugate Gradient,...).

2.1.1 Jacobi iterative method

If we consider a problem discretized using a Finite Element method, the final numerical form is usually:

$$Ax = b \tag{1}$$

We can decompose the A matrix into three matrices: U, D and L, so that:

$$A = U + D + L \tag{2}$$

For each iteration n, equation (1) becomes:

$$Ax^n = b \tag{3}$$

The Jacobi iterative method sets:

$$Dx^{n+1} = b - (U + L)x^n \tag{4}$$

and so:

$$x^{n+1} = D^{-1}b - D^{-1}(U + L)x^n \tag{5}$$

2.1.2 Gauss-Seidel iterative method

Starting from the same equation (1), the Gauss-Seidel iterative method sets:

$$(D + U)x^{n+1} = b - Lx^n \tag{6}$$

and so:

$$x^{n+1} = (D + U)^{-1}b - (D + U)^{-1}Lx^n \tag{7}$$

2.1.3 Conjugate Gradient iterative method

Beginning with the same equation (1), the Conjugate Gradient method [3] sets:

$$x^{n+1} = x^n + \alpha^n p^n \tag{8}$$

where α^n is a parameter depending on A, on the residual vector r^n (which is defined by $r^n = b - Ax^n$) and on the descent direction p^n. The parameter α^n is given by:

$$\alpha^n = \frac{{p^n}^T r^n}{{p^n}^T A p^n} \tag{9}$$

The descent direction is given by:

$$p^n = r^n + \frac{\|r^n\|^2}{\|r^{n-1}\|^2} p^{n-1} \tag{10}$$

2.2 Possible improvements for iterative solvers: preconditionners

One way to improve the convergence of the solution is to introduce preconditionners in the system [3]. Here, the preconditionner is a square matrix C. The equation (1) becomes:

$$CAx = Cb \tag{11}$$

The aim of using preconditionners is to lead the product CA as close as equal to a unity square matrix.

- For example, a simple way to find a satisfying preconditionner is to set $C = D^{-1}$, where $D = AI$ (I is the identity matrix). This preconditionner is therefore the so called diagonal preconditioner.
- For symmetric matrices, one can use the SSOR preconditionner (Symmetric Successive Over Relaxation). In this case, the matrix U which is defined in equation (2) and a relaxation parameter ω is introduced by the user ($\omega \in]0, 2[$). This preconditionner is defined by:

$$C = \frac{1}{\omega(2-\omega)}[(D - \omega U)D^{-1}(D - \omega U)^T]^{-1} \tag{12}$$

2.3 Multigrid solving principles and strategies

2.3.1 Multigrid methods

The basic principles of these solvers were set in the 1960's years and they were very improved in the 1970's by A. Brandt [1]. They were commonly developped in fluid mechanics to solve large scale problems and today, some commercial softwares still use them. They also have applications in thermics and solid mechanics. Good states of the art were made in the beginning of the 1990's by I. D. Parsons and J. F. Hall [5, 6] and P. Wesseling [7]. One recent french study led by A. Gravouil [2] partially used the multigrid principles.

2.3.2 Multigrid strategies

2.3.2.1 Multigrid basics The real disadvantage of iterative methods is the early decrease of convergence speed when solving a problem.

One way to improve iterative solvers is to consider two different meshes on a same domain: one fine mesh (or "grid") and one coarse mesh. In fact iterative solvers are quite skillful in finding high frequency variations of the solution but use more time to find low frequency variations of the solution. The mean idea is then to solve low frequency variations of the solution on a coarse mesh and high frequency variations on a fine mesh (Figure 2). This principle can be easily extended to more than two meshes.

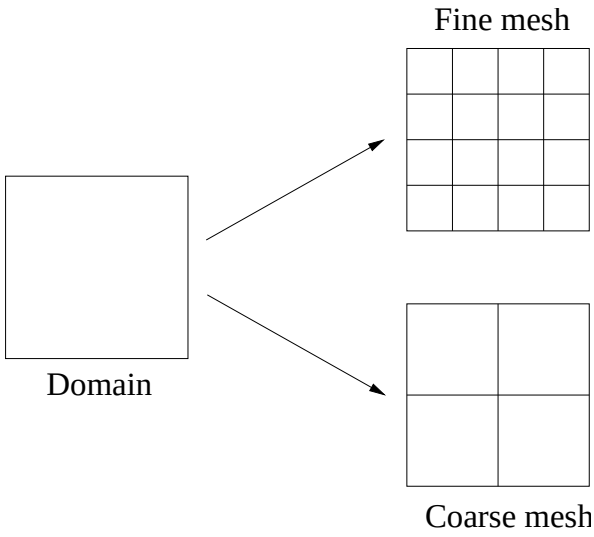

Figure 2: Two grids are superposed on the same spatial domain.

The goal is to solve the problem on the fine mesh with the help of two grids to increase the convergence. As shown in figure 3, the calculation process starts on the fine grid; after a number of iterations (generally, the solution has not converged), residuals (r_f) found on the degrees of freedom of the fine grid are reported on the nodes of the coarse mesh: this operation is called "restriction" and is realised with an operator (R) that selects common degrees of freedom. A set of iterations is then performed on the coarse grid. Results found on the coarse grid (x_c) are interpolated from the coarse grid to build an approximation of the solution vector on the fine grid: this is the "prolongation" (P).

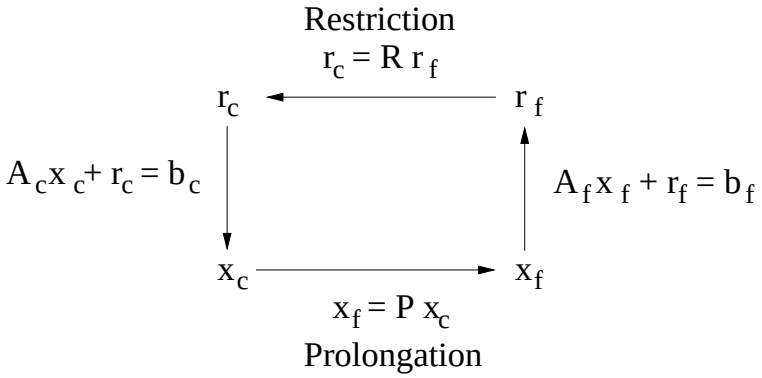

Figure 3: Restriction and prolongation between two grids

Starting from this approximation, a new set of iterations is performed on the fine grid.

This cycle is repeated until the solution has converged on the fine grid. This principle can be extended to more meshes.

2.3.2.2 Building of prolongation and restriction operators The restriction and prolongation operators are built in order to conserve energy between the meshes [2]. The prolongation operator P is first built by interpolating values from the coarse mesh to the fine mesh. To build the restriction operator R, one states the conservation of residual energy in the two meshes:

$$x_c^T r_c = x_f^T r_f \tag{13}$$

As shown in figure 3, this leads to:

$$x_c^T R r_f = x_c^T P^T r_f \tag{14}$$

and so:

$$R = P^T \tag{15}$$

2.3.2.3 Strategies for using intermediate meshes With more than two meshes related to the same domain, we can discern two strategies of changing from one grid to another one. In the first one, called "V-Cycle" , we begin iterations on the finest grid, restrict the results on the closest coarser grid, perform iterations, repeat these operations until no coarser grid remains. Then the results found on the coarsest grid are prolongated on the closest finer mesh where a set of iterations are performed; these operations are repeated upto the first grid. The solver performs as many V-Cycles as needed to reach the convergence.

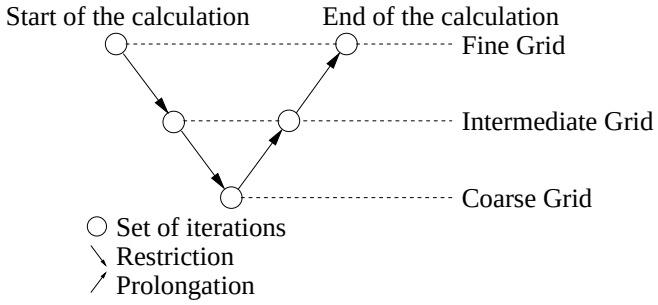

Figure 4: V-Cycles on three grids.

The second strategy, called "W-Cycles", is a variant of the V-Cycles strategy (see figure 5 in the case of three grids). As in the precedent strategy, as many "W-Cycles" as needed are performed to reach the convergence.

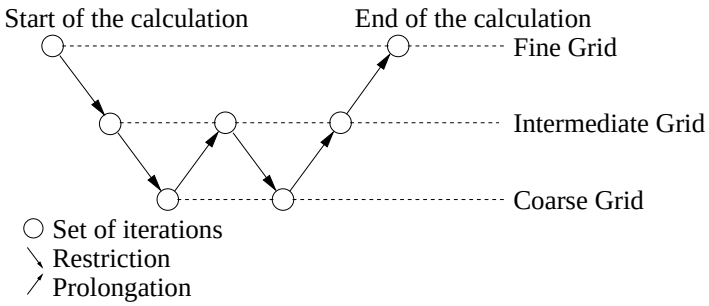

Figure 5: W-Cycles on three grids.

3 IMPLEMENTATION OF MULTIGRID METHODS IN A FINITE ELEMENT SOLVER

3.1 The FEM solver DynELA

DynELA is the C++ object oriented finite element solver developped by the Laboratoire de Génie de Production (LGP) of the Ecole Nationale d'Ingénieurs de Tarbes (ENIT) [4]. It was first developped as an implicit solver with direct resolution. This implicit solver is organized around the C++ class "Domain". It is mainly composed of a list of nodes and a list of elements. The direct implicit solver is runned on the domain (see figure 6).

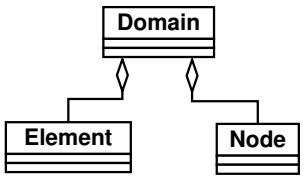

Figure 6: UML structure of C++ classes before modifications.

3.2 New implementations

We introduced a new class called "Grid". This class gets some attributes of the old "Domain" class: the list of nodes and the list of elements. The "Domain" class is modified and is now composed of a list of grids (see figure 7).

In order to perform multigrid solves with DynELA, we introduced new methods in "Domain" class. We can discern three main types of new methods that are implemented in this class:

- Iterative methods: we have implemented different iterative methods : Jacobi, Gauss-Seidel, Conjugate Gradient. These methods can be selected by the user before running the calculation.

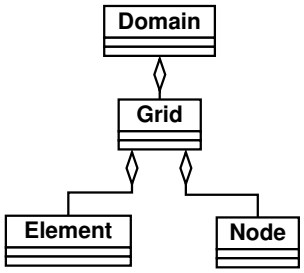

Figure 7: UML structure of C++ classes after modifications.

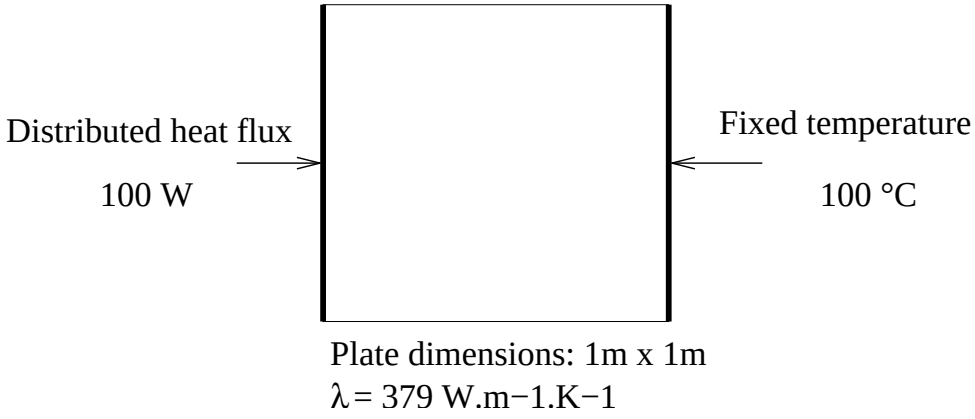

Figure 8: First test case.

- Preconditionners: we have implemented two types of preconditionners which are applied to the regular systems on each grid: Diagonal and SSOR preconditionners; in the case of the SSOR preconditionner, the value of the coefficient ω is set to 1.
- Multigrid solvers: the user can choose V-Cycles or W-Cycles strategies to solve the problem. The multigrid solvers process both thermal and mechanical 2D plane strain problems.

3.3 Examples of multigrid resolution

3.3.1 Basic test

We first tested our program on a simple case: a plate divided in thermal rectangular elements. This plate is submitted to heat flux on one side and the temperature is fixed on the opposite side (see figure 8).

The mesh has been imported and new grids have been created. We proceeded to multiple calculations to compare the skewness of the multigrid program with the direct DynELA implicit solver: different numbers of grids, different precision rate asked.

The results found on this short example agreed with the well-known analytical solution. It converges easily and we could ask precisions close to 0,001% without meeting problems on calculation.

3.3.2 Second test

We built a 2D geometry figuring a plate made of two zones with different thermal conductivities where both boundary conditions are applied: fixed temperature and repartited thermal flux (see figure 9).

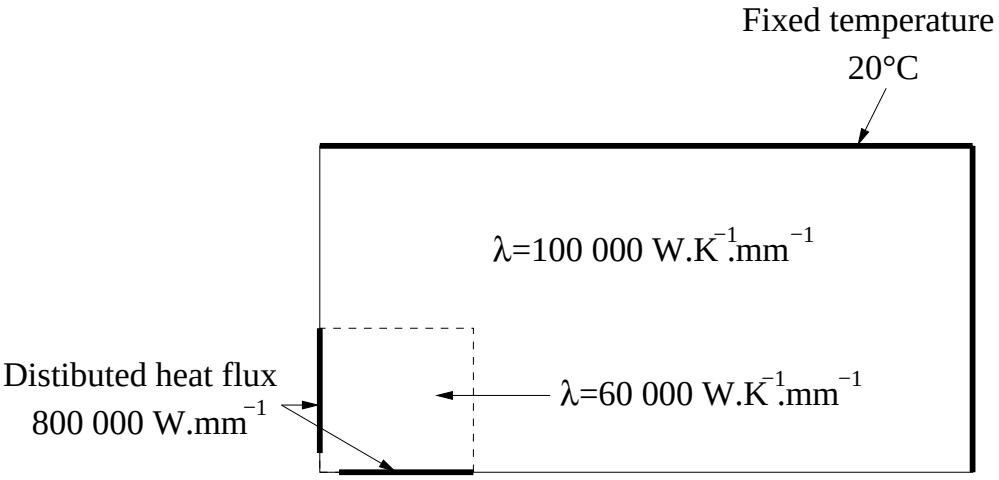

Plate dimensions: 50 mm x 25 mm

Internal zone dimensions with different thermal conductivity: 10 mm x 10 mm

Figure 9: Second test case.

The direct calculation has been done with the three following thermal implicit solvers: IDEAS, ABAQUS and DynELA implicit direct solver. The results are presented in figure 10.

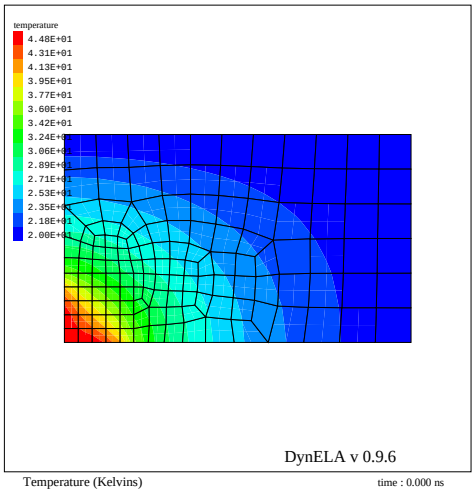

Figure 10: Distribution of temperatures in the second test case.

For the multigrid calculation, we first imported the mesh, the loads and the boundary conditions defined with IDEAS. We tested the different solvers and preconditionners implemented. As an example, we can see in table the nodal temperatures obtained in a V-Cycle resolution involving two grids, the Conjugate Gradient iterative method and the SSOR preconditionner (see table 1). For a demanded precision of 0,7% between the modules of primal and dual values vectors, the program performed 100 iterations on the two grids. We compared the results obtained with different solvers.

Node number	Node temperature obtained with implicit direct solver (DynELA, IDEAS, ABAQUS)	Node temperature obtained with DynELA multigrid solver	Difference on node temperature in %
2	44,82	44,12	-1,57
3	44,94	44,14	-1,57
6	29,50	30,09	1,98
30	39,66	39,94	0,71

Table 1: Comparison of results obtained on four nodes with multigrid and direct solvers.

The small size of the problem does not permit us to conclude about CPU costs. Moreover, the algorithm has not been optimised, so we could not compare it to commercial

direct solvers.

4 CONCLUSION

The multigrid algorithm we implemented in DynELA is currently extended to three dimensionnal thermal and mechanical fields in order to take into account the plastic behaviour in power electronics converters parts.

In fact, this multigrid method implementation constitute the first step in the building of a multidomain solver in which we could use it to perform calculations on no compatible meshes and multitime problems (both explicit and implicit subdomains together).

REFERENCES

[1] A. Brandt. *Multi-level adaptive solutions to boundary-value problems*, Mathematics of Computation, Vol. 31, Number 138, 333-390, April 1977.

[2] A. Gravouil, *Méthode multi-échelles en temps et en espace avec décomposition de domaines pour la dynamique non-linéaire dezs structures*, Thèse LMT Cachan 2000/14, 2000.

[3] P. Lascaux et Raymond Théodor, *Analyse numérique matricielle appliquée à l'art de l'ingénieur: Tome 2 - Méthodes itératives, 2ème édition*, Dunod, 1994.

[4] O. Pantalé, *An object-oriented programming of an explicit dynamics code: application to impact simulation*, Advances in Engineering Software, Vol. 33, 297-306, 2002.

[5] I. D. Parsons and J. F. Hall, *The multigrid method in solid mechanics: Part I - Algorithm description and behaviour*, International Journal for Numerical Methods in Engineering, Vol. 29, 719-737, 1990.

[6] I. D. Parsons and J. F. Hall, *The multigrid method in solid mechanics: Part II - Practical application*, International Journal for Numerical Methods in Engineering, Vol. 29, 739-753, 1990.

[7] P. Wesseling, *An introduction to multigrid methods*, John Wiley & Sons, 1991.