83

Applications of Dual Schur Complement Preconditioning to Problems in Computational Fluid Dynamics and Computational Electro-Magnetics

Quang Vinh Dinh and Thierry Fanion

1 Introduction

Domain-based parallel implementation of numerical codes using unstructured grids have been very successful for codes based on explicit integration schemes. For implicit schemes, which require successive linear solves, there is still room for improvement and research toward an efficient linear solver based on domain partitioning. In particular, since an efficient solution of the overall nonlinear problem does not require each successive linear problem to be solved to maximum accuracy, iterative methods are usually preferred.

In [Ven94], linear systems are solved by a preconditioned iterative method with a block diagonal preconditioner on interfaces and involving, within each subdomain, an incomplete factorization corresponding to a fixed sparsity pattern. While the local and parallel solves are efficient since their cost are linear in size, the convergence of the outer method degrades when the number of subdomains is high. A coarse grid solver has also been proposed in [Ven94] to alleviate this problem, at the cost of introducing a complex agglomeration procedure.

In [FMR94], a dual Schur complement method is presented for linear problems in elasticity: it is shown that the dual version of the method is preferable from a spectral convergence theory point of view. Indeed, the number of "outer" Schur iterations does not depend much on the number of subdomains into which the initial mesh has been divided. But this remarkable result is achieved with the use of direct solvers in each subdomain. Reusing previous right-hand sides at the "outer" level in reconjugation techniques also proves to be efficient: this is shown in [Rou94] for nonlinear problems in elasticity.

Ninth International Conference on Domain Decomposition Methods
Editor Petter E. Bjørstad, Magne S. Espedal and David E. Keyes

Following these lines, we would like to find a domain-partitioned linear solver which is suitable for applications in Computational Fluid Dynamics (CFD) and Computational Electro-Magnetics (CEM). We begin by describing our CFD solver; from its parallel implementation on distributed memory machines, we infer the desired characteristics of our parallel linear solver. We then present three solvers based on dual Schur complement methods and discuss their merits on representative problems. A more realistic three-dimensional result is then given to support our discussion. Lastly, we present some future work, applying these techniques to CEM problems.

2 VIRGINI: a CFD Solver for Low and High-speed Aircraft Design

VIRGINI is a two/three-dimensional Navier-Stokes solver developed at Dassault-Aviation for the last 10 years. It is extensively used for the simulation of viscous flows including modelization of turbulence phenomena and nonequilibrium air. We refer to [CMR94] and the bibliography therein for a complete description of its ingredients, which we would like now to review briefly.

Governing Equations

For the sake of simplicity, we restrict ourselves to viscous turbulent flows. Let ρ, $\boldsymbol{u}$, and E denote respectively the density, the velocity, and the total energy per unit mass of fluid. The mass-averaged Navier-Stokes equations for a compressible viscous fluid read as conservation laws in flow domain Ω for:

$$\text{mass} \qquad \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0, \tag{2.1}$$

$$\text{momentum} \qquad \frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) = \nabla \cdot \boldsymbol{\sigma}, \tag{2.2}$$

$$\text{energy} \qquad \frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho E \boldsymbol{u}) = \nabla \cdot (\boldsymbol{\sigma} \boldsymbol{u}) - \nabla \cdot \boldsymbol{q}, \tag{2.3}$$

where $\boldsymbol{\sigma}$ is the Cauchy-Reynolds shear stress tensor and $\boldsymbol{q}$ is the heat-flux vector. Appropriate boundary conditions, usually of the no-slip type, are enforced on $\partial\Omega$. Using the above set of equations to describe the mean flow, we rely on a classical Boussinesq hypothesis and the concept of eddy viscosity to make the required turbulence closure assumptions, which lead us to the following definitions for the stress tensor

$$\boldsymbol{\sigma} = (\mu^{\text{visc}} + \mu_t^{\text{visc}})\{\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^t - \frac{2}{3}\nabla \cdot \boldsymbol{u} 1\} - (p + \frac{2}{3}\rho k)1,$$

the total energy

$$E = e + \frac{1}{2}|\boldsymbol{u}|^2 + k,$$

and the heat-flux vector,

$$\boldsymbol{q} = -(\mu^{\text{visc}}\frac{\gamma}{P_r} + \mu_t^{\text{visc}}\frac{\gamma}{P_{rt}})\nabla e.$$

Here μ^{visc} is the molecular viscosity, μ_t^{visc} is the eddy viscosity, 1 is the identity tensor, k is the turbulent kinetic energy, $\gamma = c_p/c_v$ is the ratio of the fluid specific heats. The laminar Prandtl number is taken as $P_r = 0.72$ and the turbulent Prandtl number is taken as $P_{rt} = 0.9$. Internal energy e is defined by $e = c_v T$ and pressure p is calculated from the thermodynamic state equation $p = p(\rho, T)$.

The turbulence model used belongs to the $k-\epsilon$ family (see [CMR94]) and introduces two extra equations:

$$\rho\frac{\partial k}{\partial t} + \rho\boldsymbol{u}\cdot\nabla k - \nabla\cdot((\mu^{\text{visc}} + \frac{\mu_t^{\text{visc}}}{\sigma_k})\nabla k) = H_k(k,\epsilon,\rho,\boldsymbol{u},\mu_t^{\text{visc}}) \tag{2.4}$$

$$\rho\frac{\partial \epsilon}{\partial t} + \rho\boldsymbol{u}\cdot\nabla \epsilon - \nabla\cdot((\mu^{\text{visc}} + \frac{\mu_t^{\text{visc}}}{\sigma_\epsilon})\nabla \epsilon) = H_\epsilon(k,\epsilon,\rho,\boldsymbol{u},\mu_t^{\text{visc}}) \tag{2.5}$$

which are solved for k and ϵ. The eddy viscosity is then defined as:

$$\mu_t^{\text{visc}} = \rho C_\mu \frac{k^2}{\epsilon}.$$

Here $\sigma_k, \sigma_\epsilon, C_\mu$ are modelling constants and H_k, H_ϵ are source terms.

Numerical Approximation

Different numerical approximations are used for the two systems of equations above.

For system (2.4-2.5), corresponding to the turbulence model, the positivity of k and ϵ is achieved by the combination of two main features (see [CMR94]): the use of a monotone advective finite-volume scheme and the time discretization of the source terms. This is done via a semi-implicit time-marching algorithm, leading to two decoupled linear systems to be solved at each time iteration:

$$\rho\frac{\Delta i}{\Delta t} + \rho\boldsymbol{u}\cdot\nabla i - \nabla\cdot((\mu^{\text{visc}} + \frac{\mu_t^{\text{visc}}}{\sigma_i})\nabla i) = H_i, \tag{2.6}$$

where $i = k$ or ϵ, and Δi (resp. Δt) is the variable (resp. time) increment.

In system (2.1-2.3) representing the mean flow equations (see [CMR94]), a Galerkin/least-squares finite element formulation is applied to the compressible Navier-Stokes equations which has been rewritten in the form of a symmetric advective-diffusive system in terms of entropy variables $\boldsymbol{V}^T = \partial\mathcal{H}/\partial\boldsymbol{U}$, where $\boldsymbol{U}^T = \{\rho, \rho\boldsymbol{u}, \rho E\}$ are the conservative variables and $\mathcal{H}(\boldsymbol{U}) = -\rho s$ is the generalized entropy function, with s being the physical entropy per unit mass. A fully implicit time-marching procedure is used, so that a nonlinear problem is solved at each time step. A linearization through a truncated Taylor series expansion is then performed, leading to the following linear system to be solved for variable increment $\Delta\boldsymbol{V}$:

$$\widetilde{\boldsymbol{A}}_0\frac{\Delta\boldsymbol{V}}{\Delta t} + \widetilde{\boldsymbol{A}}\cdot\nabla\Delta\boldsymbol{V} = \nabla\cdot(\widetilde{\boldsymbol{K}}\nabla\Delta\boldsymbol{V}) + \mathcal{F} \tag{2.7}$$

in which

$\widetilde{\boldsymbol{A}}_0$ is symmetric and positive definite,

$\widetilde{\boldsymbol{A}} = \{\widetilde{\boldsymbol{A}}_i\}, 1 \leq i \leq 5$ with $\widetilde{\boldsymbol{A}}_i$ symmetric,

$\widetilde{\boldsymbol{K}}$ is symmetric and positive semi-definite,

$\mathcal{F}$ is the current right-hand side.

Finally, the discretized mean flow equations and the turbulence equations are coupled through a splitting method. At a current time step, we solve the Navier-Stokes equations using turbulence data evaluated at the previous time while the turbulence equations are solved using the flow variables computed at the previous time.

All space discretizations are done on unstructured meshes with piecewise linear interpolation on tetrahedra. Since we are using VIRGINI to get steady-state solutions, time accuracy is not mandatory. Thus, the above linear systems need not be solved to maximum accuracy and our preferred linear solver is an iterative method, namely GMRES (see [BBC$^+$94]), with diagonal preconditioning for (2.6) and nodal block-diagonal preconditioning for (2.7).

Computer Implementation

An iterative linear solver like GMRES requires only two types of operations (see [BBC$^+$94]): vector inner-product and matrix-vector product. This allows us to implement in VIRGINI the so-called "matrix-free" procedure, that is matrices for systems (2.6) and (2.7) are never stored, only procedures to compute the corresponding matrix-vector products are created. This feature, along with the usage of diagonal preconditioning, give an "explicit-like" behavior to the code, which facilitates vectorization via coloring techniques and parallelization via domain partitioning.

VIRGINI has been ported on various platforms: IBM ES-9000, Convex C2-C3, NEC-SX3, Intel iPSC-860, IBM SP2, workstations (IBM RS6000, SGI), etc. The memory requirement is about 2.7KB per mesh node, which means that, on our 16-processor IBM SP2 with "thin" nodes (i.e., with 128MB of local memory), we can accommodate a "maximum-size" mesh of 750,000 nodes. We give some typical convergence data in the following simplified flowchart of VIRGINI:

Begin time-marching loop

step 1: Solve (2.7) with previous turbulence data
→ convergence level required: 10^{-1} to 10^{-3}
→ number of GMRES iterations: at most 10

step 2: Solve (2.6) with previous flow data
→ convergence level required: 10^{-3} to 10^{-5}
→ number of GMRES iterations: at most 20

End time-marching loop if nonlinear residual is small enough

Improving Performance

We would like to improve the convergence of the linear solvers while retaining, as much as possible, the nice features of VIRGINI.

One usual way is to replace the diagonal preconditioner by a more elaborate one built from an incomplete LDU factorization of the linear operators. This preconditioner introduces some extra memory and computational burdens which must be carefully evaluated.

For system (2.7), preliminary tests have shown that it requires too much extra computation and, in particular too much extra memory, which cannot be accounted for in light of the low convergence requirement. The situation is different for system (2.6) which must be solved more accurately. But then, what becomes of the incomplete LDU preconditioner in a parallel framework involving domain partitioning?

An "ad hoc" local incomplete LDU preconditioner has been proposed in [Ven94], where Dirichlet type conditions are imposed on interfaces, allowing the decoupling of the original operator into local ones. This has been shown to work well for Euler solvers, but the convergence degrades as the number of subdomains increases.

We would like now to derive, in the framework of dual Schur complement methods, a parallel solver for system (2.6) based on local incomplete LDU factorization, which, ideally, should have the following characteristics:

- Convergence rate nearly independent of the number of subdomains, even at low levels of convergence
- Local solvers, the costs of which, both in computer time and in memory requirement, are linear in problem size.
- Improvement upon a global GMRES solver with diagonal preconditioning.

In what follows, incomplete LDU factorization, iLDU in short, will be understood to be with the sparsity pattern of the original matrix.

3 Dual Schur Complement Solvers

In this section, we first present three parallel solvers based on dual Schur complement methods and discuss their merits on two model problems which are representative of VIRGINI:

problem P1) two-dimensional flow around a NACA0012 airfoil at free-stream conditions: Mach number = 0.799, incidence = 2.26°, Reynolds number $= 9 \times 10^6$. Mesh sizes are: 8008 nodes and 15794 triangles. The converged solution needs about 1000 time steps. As a model problem, we pick out the linear problem corresponding to system (2.6) at the 700th time step, where the flow is fully developed.

problem P2) three-dimensional flow over a blunt body at free-stream conditions: Mach number = 3.0, no incidence, Reynolds number $= 10^6$. Mesh sizes are: 3506 nodes and 13280 tetrahedra. We have chosen the linear turbulence problem at the 10th time-step, at the beginning of the convergence which takes about 200 time steps.

Secondly, a more realistic test case is presented in which the "best" solver is implemented.

Solver 0: Direct Local Solver

In [FMR94], a dual Schur complement method, named FETI (for Finite Element Tearing and Interconnecting), is presented for elliptic problems in elasticity. We briefly recall the ingredients of FETI, adapted to system (2.6).

In flow domain Ω discretized by an unstructured mesh (identified in the sequel with the flow domain), let A be the matrix of system (2.6), f the right-hand side vector and u the vector of unknowns. We partition Ω into N_s subdomains $\{\Omega^s\}_{1\leq s\leq N_s}$, and we denote by A^s, f^s, u^s and B^s, respectively, the subdomain discretization of operator A, the subdomain right-hand side and unknowns, and the signed matrix with entries $-1, 0, +1$ describing the subdomain interconnectivity. The original problem $Au = f$ is shown to be equivalent to the following one:

$$\forall s, 1 \leq s \leq N_s, \quad A^s u^s + B^{s^T} \lambda = f^s \tag{3.8}$$

$$\sum_{s=1}^{N_s} B^s u^s = 0, \tag{3.9}$$

where λ is the Lagrange multiplier for constraint (3.9). As in the original FETI algorithm, the local systems (3.8) are solved by a direct method and there is an "outer" iterative procedure to compute λ: since A is nonsymmetric, we have used a GCR algorithm (see [BBC$^+$94]).

We have implemented this solver for problem P1 with different mesh partitions.

Table 1 CPU performance for solver 0

Number of subdomains	1	4	8	16	32
ideal iLDU	1.0	0.25000	0.12500	0.06250	0.03125
Solver 0	not rel.	0.03900	0.01400	0.01620	0.01740

The results are shown in table 1, where we have taken as CPU time unit, the CPU time taken by a global GMRES algorithm with iLDU preconditioning, converged to 10^{-16}, on the whole mesh Ω. The other entries in the "ideal iLDU" line have been computed assuming that this GMRES algorithm has been somehow "perfectly" parallelized. Values in the "Solver 0" line are actual measures.

For different values of N^s, solver 0 performs better than the "ideal" iLDU preconditioner, although for $N^s \geq 8$ there is a performance stagnation due to the importance of communication versus computation. However, these results were obtained for a level of convergence up to machine precision, which is required for direct methods. Moreover, these direct methods entail extra burdens at the subdomain level: $O(n_s^{2.5})$ in computation and $O(n_s^{1.5})$ in memory, where n_s is the size of the local matrix A^s.

Solver 1: Iterative Local Solver

To make the extra work at the subdomain level, *linear* in size, one simple idea is to replace the local direct solvers by local iterative solvers. We have done this in solver 1, where the local solver is a GMRES algorithm with iLDU preconditioning. But now, we have to monitor two levels of convergence: ε_{glo} for the "outer" GCR algorithm and ε_{loc} for the local GMRES algorithm.

For problem P2, we have made a study of the relationship between ε_{glo} and ε_{loc} for two mesh partitions.

Table 2 Level of convergence ("outer" iterations count) for solver 1

2 subdomains	$\varepsilon_{glo} = 10^{-3}$	$\varepsilon_{glo} = 10^{-5}$	$\varepsilon_{glo} = 10^{-10}$
$\varepsilon_{loc} = 10^{-3}$	0.80×10^{-3}(8)	0.68×10^{-3}(19)	0.68×10^{-3} (44)
$\varepsilon_{loc} = 10^{-5}$	0.39×10^{-3}(8)	0.33×10^{-5}(19)	0.80×10^{-6} (44)
$\varepsilon_{loc} = 10^{-10}$	0.39×10^{-3}(8)	0.32×10^{-5}(19)	0.23×10^{-10}(44)

4 subdomains	$\varepsilon_{glo} = 10^{-3}$	$\varepsilon_{glo} = 10^{-5}$	$\varepsilon_{glo} = 10^{-10}$
$\varepsilon_{loc} = 10^{-3}$	0.72×10^{-3}(17)	0.66×10^{-3}(31)	0.66×10^{-3} (61)
$\varepsilon_{loc} = 10^{-5}$	0.29×10^{-3}(17)	0.71×10^{-5}(31)	0.63×10^{-6} (61)
$\varepsilon_{loc} = 10^{-10}$	0.29×10^{-3}(17)	0.35×10^{-5}(31)	0.42×10^{-10}(61)

The results are shown in table 2. As expected, it does not pay to converge more, at the subdomain level, than to the level of convergence fixed for the "outer" GCR algorithm: indeed, one should take $\varepsilon_{loc} = \varepsilon_{glo}$. Another interesting remark is that the higher this level of convergence is, the weaker the dependence of the rate of convergence on the number of subdomains, in a relative sense.

Solver 2: Direct Local Solver for Approximate Operator

In solver 1, we have seen that the scalability in the number of subdomains is dependent on the level of convergence required, local as well as global. This constraint is removed if, as in solver 0, the local solver is direct, but we have seen that this incurs too high a memory requirement. On the other hand, in the dual Schur formulation (3.8-3.9), the global operator is defined solely by its local representation. From the local incomplete LDU factorizations:

$$\forall s, 1 \leq s \leq N_s, \quad A^s \approx \widetilde{L}^s \widetilde{D}^s \widetilde{U}^s \quad = \quad \widetilde{A}^s, \tag{3.10}$$

we can define an operator $\widetilde{A}$ from its local contributions $\widetilde{A}^s$. Since the incomplete factorizations are done with the same sparsity pattern as for A^s, the extra work to compute and store $\widetilde{A}$ is linear in n_s and the local solvers are direct. Thus the application of solver 0 to $\widetilde{A}$ will result in a parallel solver having all the required characteristics.

If $\widetilde{A}$ is a good "approximation" to A, we can now propose solver 2 which has the following ingredients:

- global iterative algorithm: GMRES with level of convergence ε_{glo}
- global preconditioner: solver 0 applied to $\widetilde{A}$ with the same level of convergence

Table 3 Level of convergence ("outer" iteration count) for solver 2

	$\varepsilon_{glo} = 10^{-3}$	$\varepsilon_{glo} = 10^{-5}$	$\varepsilon_{glo} = 10^{-10}$
2 subdomains	$0.57 \times 10^{-4}(9)$	$0.44 \times 10^{-6}(12)$	$0.36 \times 10^{-11}(21)$
4 subdomains	$0.44 \times 10^{-4}(9)$	$0.41 \times 10^{-6}(12)$	$0.36 \times 10^{-11}(21)$

We have implemented solver 2 for problem 2 and results shown in table 3 suggest that we have come up with a good solution since the "outer" iteration count does not vary when we go from 2 to 4 subdomains.

A Realistic Test Case

To support our discussion, we have run VIRGINI, with solver 2 implemented for system (2.6), on a more realistic flow simulation: a low-speed flow around the forebody of a military aircraft, namely a Mirage 2000, at high angle of attack.

Free stream conditions were: Mach number = 0.2, incidence = 50°, altitude = 3000 meters. Mesh sizes were around 50,000 nodes and 275,000 elements.

We have made a thorough comparison, during the first 100 time-steps, between this version of VIRGINI and the original version. The gains, for a convergence level for system (2.6) fixed at 10^{-5}, were:

in iteration count $\approx 60\%$

in CPU time $\approx 4\%$.

The smaller gain in CPU time can be accounted for by the extra local incomplete factorizations done at each time step. Different operator "freezing" strategies should be used here to alleviate this problem.

The complete convergence for this simulation needed about $2,000$ iterations and over 33 hours on a IBM SP2 with 4 (thin) processors.

4 SPECTRE: a CEM Solver for Aircraft Design

We would like now to sketch some future work applying dual Schur complement methods to CEM problems. SPECTRE is a three-dimensional solver for the Maxwell equations developed at Dassault-Aviation for the last 6 years. We refer to [CLL$^+$90] and [CZJ96] as well as the bibliography therein for a complete description of its ingredients, which we would like now to review briefly.

Governing Equations

Let $\boldsymbol{H}, \boldsymbol{E}, \boldsymbol{K}, \boldsymbol{J}$ denote respectively the magnetic field, the electric field, the magnetic surface current and the electric surface current. SPECTRE solves the time-harmonic Maxwell equations for perfectly conducting material which read, for domain Ω:

$$\begin{aligned}
\nabla \times \nabla \times \boldsymbol{H} - k^2 \boldsymbol{H} &= 0 \quad \text{in} \quad \Omega, & (4.11)\\
\nabla \times \nabla \times \boldsymbol{E} - k^2 \boldsymbol{E} &= 0 \quad \text{in} \quad \Omega, & (4.12)\\
-\boldsymbol{n} \times \boldsymbol{E} &= \boldsymbol{K} \quad \text{on} \quad \partial\Omega, & (4.13)\\
\boldsymbol{n} \times \boldsymbol{H} &= \boldsymbol{J} \quad \text{on} \quad \partial\Omega, & (4.14)
\end{aligned}$$

where $\boldsymbol{n}$ is the outward unit normal to $\partial\Omega$ and k is the incident wave number. In the far field, the Sommerfeld radiation condition should also be enforced.

Numerical Approximation

This radiation condition constitutes a major numerical difficulty in Maxwell equations. In the CEM community, it is customary to distinguish two types of problems:

interior problems: such as simulating fields in waveguides and cavities; there is no need for a radiation condition.

exterior problems: such as simulating fields scattered or radiated from structures; the Sommerfeld condition must be enforced.

For aircraft design, the above two situations are present. As proposed in [CZJ96], SPECTRE partitions domain Ω into two parts separated by a bounding surface and uses a coupling method between:

EFIE: an Electric Field Integral Equation solution, *exterior to the bounding surface*, that exactly enforces the Sommerfeld radiation condition. Piecewise linear electric surface currents $\boldsymbol{J}$ are defined on the boundary discretized by an unstructured surface mesh composed of triangles, leading to the following equation:

$$\boldsymbol{Z}_J \boldsymbol{J} = \boldsymbol{V}_i \qquad (4.15)$$

where $\boldsymbol{Z}_J$ is a dense matrix and $\boldsymbol{V}_i$ represents the incident field. A direct Gauss solver is used for the solution of (4.15).

MFVE: a Magnetic Field Volumic Equation solution, *interior to the bounding surface.* For a given electric field $\boldsymbol{E}$ computed from (4.15), equation (4.11) along with its natural boundary condition (4.13) is solved for magnetic field $\boldsymbol{H}$. A finite element formulation is used with an unstructured volumetric mesh composed of tetrahedra; it is based on the H(rot) tetrahedral element (see [CZJ96]) with unknowns defined on *edges.* The resulting linear system, with a sparse symmetric and complex matrix, is solved by a QMR iterative procedure with a SSOR preconditioner (see [FN91] and [BBC$^+$94]).

The above coupling is done in a *direct* manner. The bounding interface unknowns are eliminated by a succession of *independent* linear solves for MFVE, one for each degree of freedom on this interface.

MFVE: Parallel Implementation using Dual Schur Methods

For these solutions, we would like to apply the different parallel solvers defined in the previous section. In this perspective, we need to keep in mind the following differences with the CFD case:

1. The level of convergence required is higher: 10^{-8}.
2. Preliminary tests have shown that, due to the particular spectrum of the operator, the "outer" GCR algorithm in the dual Schur methods should be replaced by a QMR procedure.
3. We are solving for a given linear operator with a large number of independent right-hand sides: reconjugation techniques proposed in [Rou94] should be useful.

In light of these remarks, solver 0 seems to have the advantage.

5 Conclusion

We have proposed a parallel preconditioner, based on dual Schur methods and incomplete LDU factorization, which seems to be scalable both in terms of the number of subdomains as in terms of the level of convergence required.

Applications to problems in CFD have given thus far only a small gain compared to the usual diagonal preconditioner. We are expecting higher gains as we go on to "stiffer" problems such as those encountered in unsteady flows or multi-disciplinary optimization.

Applications to problems in CEM are still in the development stage.

REFERENCES

[BBC+94] Barret R., Berry M., Chan T., Demmel J., Donato J., Dongarraa J., Eijkhout V., Pozo R., Romine C., and van der Vorst H. (1994) *Templates for the solution of linear systems : building blocks for iterative methods*. SIAM.

[CLL+90] Calnibalosky C., Leflour G., Lohat P., Lombard J., Quadri J., and Vukadinovic N. (November 1990) Electromagnetic calculation of a whole aircraft by the code spectre. In *Proc. of JINA 90*, pages 83–87. CNET-IEEE, Nice.

[CMR94] Chalot F., Mallet M., and Ravachol M. (January 1994) A comprehensive finite element navier-stokes solver for low and high-speed aircraft design. *AIAA 94-814* .

[CZJ96] Cwik T., Zuffada C., and Jamnejad V. (April 1996) Modeling three-dimensional scatterers using a coupled finite element-integral equation formulation. *IEEE Trans. Antennas and Propagat.* 44(4): 453–459.

[FMR94] Farhat C., Mandel J., and Roux F. (1994) Optimal convergence properties of the finite element tearing and interconnecting domain decomposition method. *Comp. Meths. Appl. Mech. Eng.* (115): 365–385.

[FN91] Freund R. and Nachtigal N. (1991) A quasi-minimal residual method for non-hermitian linear systems. *Numerische Mathematik* 60: 315–339.

[Rou94] Roux F. (April 1994) Parallel implementation of a domain-decomposition method for non-linear elasticity problems. In D.Keyes Y.Saad D. (ed) *Proc.of the Workshop on Domain-based Parallelism and Problem Decomposition methods in Comp. Sci. and Eng.*, pages 161–175. SIAM, Minneapolis.

[Ven94] Venkatakrishnan V. (January 1994) Parallel implicit unstructured euler solvers. *AIAA J.* 32(10): 1985–1991.