A Finite Element Method for Virtual Reality Data

Stéphane Del Pino, Erkki Heikkola,
Olivier Pironneau and Jari Toivanen

April 20, 2000

Abstract

Image synthesis languages like `POV-Ray` and VRML do not give detailed description of the surfaces of the objects they generate, and therefore it is quite difficult to use them for engineering computations. Rather than trying to solve the problem of data translation, we propose to apply the fictitious domain method in the numerical solution. We explain what are the difficulties and give some solutions with an illustration of a finite element computation for a three-dimensional acoustic scattering problem.

Une méthode d'éléments finis pour des structures de données de la Réalité Virtuelle.

Résumé en français

Les languages de la synthèse d'image comme `POV-Ray` et VRML ne donnent pas une description détaillée des surfaces des objets générés; il est donc difficile de les utiliser pour des calculs d'ingénierie. Plutôt que d'essayer de résoudre le difficile (mais technique) problème de la traduction de donnée afin d'obtenir un maillage de surface admissible puis d'appeler un mailleur automatique, nous proposons une solution par la méthode des domaines fictifs. Nous montrons aussi où sont les difficultés et nous proposons des solutions avec une illustration par un calcul éléments finis sur un problème d'acoustique tridimensionel de grande taille.

1 Introduction

The algorithms of Virtual Reality (VR)[1][8] have to compromise between realistic rendering and speed.

Constructive Solid Geometry is quite popular in VR and consequently the domain of integration of the PDE is not given by its boundary but by set operations on simple elementary volumes, typically unions and intersections and sometimes extrusion. Consider for example the following scene (see figure 1) described in the `POV-Ray` language:

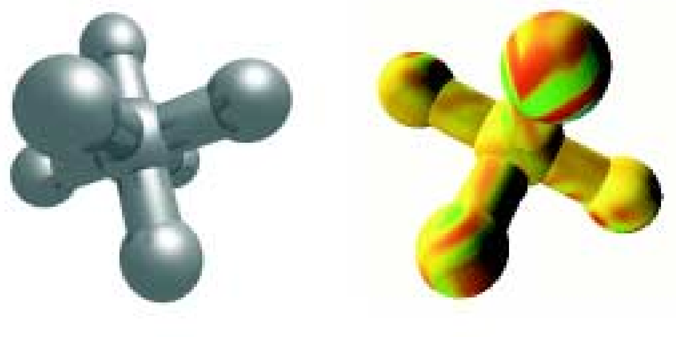

Figure 1: **Left***A scene displayed by* `POV-Ray`*. The objects are never intersected, it is the graphic rendering that takes care of the problem.* **Right** *The trace of the real part of the scattered acoustic field on the surface of the geometry.*

```
#declare altere = union{ cylinder {<-1.5,0,0> <1.5,0,0>, .35}
             sphere {<-1.5,0,0>, 0.5} sphere {<1.5,0,0>, .5}}
union { object { altere rotate z*90 } object {altere scale 1}
        object { altere rotate y*90 } sphere{<0,0,0>, 0.6 } }
```

It is a set of spheres and cylinders with some realistic texture; notice that there is no information about their intersection. This is seen only in the rendering phase which is based on the zbuffer algorithm (zbuff[] is initially filled with large values):

```
for(t=0;tmax;t++) if(z[t] <= zbuff[x[t]][y[t]])
          { putpixel(x[t],y[t],z[t]); zbuff[x[t]][y[t]]=z[t];}
```

This C-program displays 3D objects $\{t \to (x[t], y[t], z[t])\}$ on a screen of size hmax$\times$vmax. By displaying all objects this way each new voxel (3D point) of an object lights a pixel (2D point) only if it is in front of all previously displayed voxels that fall on the same pixel.

Engineering data on the other hand use Bezier or B–spline patches from which it is easier to derive a triangulation of the surfaces in the scenes, a necessary step for the generation of three-dimensional meshes.

The problem that this note addresses is the solution of a Partial Differential Equation (PDE) in a domain given by set operations on a large number of elementary shapes such as spheres, cones, cylinders and boxes.

In a series of notes, J.L. Lions [5],[6],[7] and one of the authors have shown that CSG opens new vistae for the speed up of computations by Domain Decomposition. We will use these results also in a subsequent work while here we simply solve the direct problem in three dimensions.

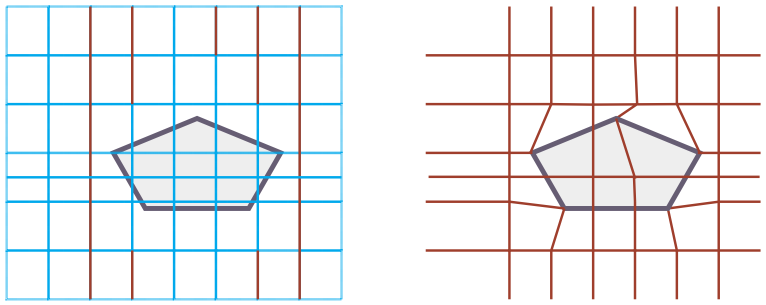

Figure 2: *A locally fitted quadrilateral mesh: the nearby vertices are moved on the boundary of the polygonal domain so that the quadrangulation becomes admissible.*

2 Three-dimensional Helmholtz equation

We consider the scattering of a time-harmonic acoustic plane wave $\exp(i\boldsymbol{\omega}\cdot x)$ by a geometry created using the CSG of the `POV-Ray` program (figure 1). Such scattering can be modeled by the three-dimensional Helmholtz equation with an absorbing boundary condition on a rectangular artificial boundary. Here, we use an impedance boundary condition on the surface of the geometry:

$$\begin{aligned} -\Delta u(x) - \omega^2 u(x) &= 0, \qquad && x \in \Pi \setminus \bar{D}, \\ \frac{\partial u(x)}{\partial n} - i\omega u(x) &= g(x), && x \in \partial D, \\ \mathcal{B}u(x) &= 0, && x \in \partial \Pi. \end{aligned} \tag{1}$$

We denote the obstacle by D and the surrounding rectangle by Π. The parameter ω is the wave number, which determines the wavelength of the scattered wave, while the operator $\mathcal{B}$ corresponds to the chosen absorbing boundary condition.

3 Locally fitted meshes

We apply the piecewise linear finite element method with mass lumping to obtain a linear system $\mathbf{Au} = \mathbf{f}$ corresponding to the equation (1). We need to construct a tetrahedral mesh into the computational domain $\Pi \setminus D$, and, for this purpose, we use locally perturbed orthogonal meshes called locally fitted meshes [4, 9]. The use of such meshes is motivated by the fact that they allow us to realize the GMRES method in a low-dimensional subspace [2, 4].

Locally fitted meshes are obtained by first creating an orthogonal grid Π_h for the domain Π. We modify Π_h such that the nodes, which are neighboring ∂D are shifted locally onto the boundary of D. This perturbation changes the grid cells only in the neighborhood of the surface and results in a deformed grid $\tilde{\Pi}_h$. The old orthogonal mesh will be used later to construct a preconditioner. So we require that there is a one-to-one correspondence between the nodes of Π_h

and $\tilde{\Pi}_h$. The modified cells are then divided into tetrahedrons according to the shape of the obstacle, which leads to a second-order approximation for smooth boundaries.

The practical realization of the generator of 3D locally fitted meshes requires the user to provide a C–function, which tells if a certain coordinate x is inside or outside of the geometry D. Such a characteristic function is easily constructed for a geometry created by the `POV-Ray` program.

4 Algebraic fictitious domain method

The solution of the linear system thus constructed is a formidable task because the number of mesh point is large. Following Kuznetsov[4] we use an algebraic fictitious domain method. It is based on the idea of replacing the linear system $Au = f$ resulting from the discretization on the admissible mesh by an equivalent, but enlarged system $\tilde{A}\tilde{u} = \tilde{f}$ for the values of u on all the vertices of Π not just those of $\Pi \backslash D$. The benefit of this approach is that there are wider possibilities to construct efficient preconditioners for the enlarged system than for the original one. For instance the finite difference scheme on the original orthogonal mesh of Π gives a matrix B which is a good preconditoner for $\tilde{A}$; then an iterative scheme like

$$B(\tilde{u}^{m+1} - \tilde{u}^m) = \tilde{f} - \tilde{A}\tilde{u}^m$$

is very fast. A detailed description of the implementation of the fictitious domain solver for the 3D Helmholtz equation is given in [2].

5 Interfacing Virtual Reality Data

To use the *virtual reality* description of the computational domain (figure 1 and the program of section 1), one has to generate a characteristic function for the object D.

The construction of this function was done in two steps, first by writing an interpreter for the language describing the scene and then by constructing the characteristic functions of all the elementary objects of the scene.

The *virtual reality* language chosen was `POV-Ray`[8] because it is stable and portable but it can also be done with `vrml` as easily. Also `VRML` just implements the set union but no set intersection. Moreover, one can use it as a visualization tool thanks to the iso-surface patch of Suzuki[10].

To parse the program of Section 1 is easy but `POV-Ray` has also conditional statements, loops, symbolic variables, functions, etc.; so we have used `lex` and `yacc` because both of them generate `C` or `C++` parsers.

In the `C++` produced, an abstract `Shape` class is defined; it provides a standard interface to every kind of shapes. Its children are specialization such as `Sphere` or `Cube` (primitives), but also boolean operations such as `Union` and

`Intersection`. The same kind of design is used for geometrical transformations of `POV-Ray`, (`scale`, `rotate` or `translate`).

As every primitive shape is simple (sphere, cube, cone, ...), one can easily associate to it its characteristic function. This is done by specializing the virtual function of the class `Shape` for each of them. To know if a vertex is in an *object* one has to compute the inverse of each transformation associated to it and then check if the antecedent is in the primitive `Shape`.

This implementation makes the formulation very close to the mathematics, and as the results of boolean operations are shapes, complex objects descriptions and their characteristic function evaluation may be recursive. There is a problem however with two dimensional structures like polygonal facets.

6 Numerical results

We used an efficient parallel implementation of the method introduced in Heikkola et al[3], which enables us to solve large-scale 3D scattering problems in which the number of wavelengths along the geometry may be up to 100. The computation was performed in a Origin 2000 parallel computer.

In our test case, the wavelength was $\lambda = 1$ and the rectangle enclosing the geometry was given by the two corners $(-3, -3, -3)$ and $(3, 3, 3)$. The object's diameter is 4. The locally fitted mesh was created such that there were 20 mesh nodes per wavelength, and we computed the final solution values on each mesh node in the rectangle Π (121^3 complex numbers). Thus, by linear interpolation, we obtained a numerical solution for the whole computational domain.

There were 8856 nodes on the surface of the geometry, while the iterations were performed in a subspace of dimension 40740. The dimension of the total system was 121*121*121=1.8 million nodes. Sufficient accuracy was reached with 64 GMRES iterations. The wall clock time for the iterative solution with 16 processors of SGI Origin 2000 was 62 seconds. Various initializations took 30 seconds.

References

[1] J. Hartman and J. Wernecke, "The VRML 2.0 Handbook" Addison-Wesley 1996.

[2] E. Heikkola, Y. Kuznetsov, and K. Lipnikov, "Fictitious domain methods for the numerical solution of three-dimensional acoustic scattering problems", *J. Comput. Acoustics* **7** (1999) 161–183.

[3] E. Heikkola, T. Rossi, and J. Toivanen, "Efficient iterative solution of high frequency acoustic scattering problems", in Proceedings of the Third SIAM Conference on the Numerical Aspects of Wave Propagation, SIAM, 2000. To appear.

[4] Y. Kuznetsov, and K. Lipnikov, "3D Helmholtz wave equation by fictitious domain method", *Russian J. Numer. Anal. Math. Modelling* **13** (1998) 371–387.

[5] J.L. Lions, O. Pironneau, Algorithmes parallèles pour la solution de problèmes aux limites, C.R.A.S., 327, pp 947-352, Paris 1998.

[6] J.L. Lions, O. Pironneau, Sur le contrôle des sytèmes distribués. C.R.A.S., 327, pp 993-998, Paris 1998.

[7] J.L. Lions, O. Pironneau, Domain decomposition methods for CAD. C.R.A.S., 328, pp 73-80, Paris 1999.

[8] A. Wardley, Persistence of Vision, `POV-Ray` in `http://www.povray.org/`.

[9] A. Supalov, "Development of algorithms for generating 3D locally fitted grids and their program realization", Ph.D. thesis, Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow, 1994. In Russian.

[10] R. Suzuki, A patch to `POV-Ray` for iso-surfaces. In `http://www.public.usit.net/rsuzuki/e/povray/iso/index.html`.

Authors addresses

1. Stephane Del Pino (`delpino@ann.jussieu.fr`) et
   Olivier Pironneau (`pironneau@ann.jussieu.fr`), Laboratoire d'analyse numerique, université Paris VI, 175 rue du Chevaleret, F-75013 Paris.

2. Erkki Heikkola (`emsh@mit.jyu.fi`) et
   Jari Toivanen (`tene@mit.jyu.fi`), Department of Mathematical Information Technology University of Jyväskylä, P.O. Box 35, 40351 Jyväskylä, Finland.