Computer Physics Communications 200 (2016) 291–299

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

VOFI — A library to initialize the volume fraction scalar field☆

S. Bnà [a], S. Manservisi [a], R. Scardovelli [a,*], P. Yecko [b], S. Zaleski [c,d]

[a] DIN – Lab. di Montecuccolino, Università di Bologna, Via dei Colli 16, 40136 Bologna, Italy
[b] Physics Department, Cooper Union, New York, NY, USA
[c] Sorbonne Universités, UPMC Univ Paris 06, UMR 7190, Institut Jean Le Rond d'Alembert, F-75005, Paris, France
[d] CNRS, UMR 7190, Institut Jean Le Rond d'Alembert, F-75005, Paris, France

ARTICLE INFO

Article history:
Received 8 July 2014
Received in revised form
27 July 2015
Accepted 28 October 2015
Available online 12 November 2015

Keywords:
Implicit functions
Numerical integration
Volume fraction function
VOF method

ABSTRACT

The VOFI library has been developed to accurately calculate the volume fraction field demarcated by implicitly-defined fluid interfaces in Cartesian grids with cubic cells. The method enlists a number of algorithms to compute the integration limits and the local height function, that is the integrand of a double Gauss–Legendre integration with a variable number of nodes. Tests in two and three dimensions are presented to demonstrate the accuracy of the method and are provided in the software distribution with C/C++ and FORTRAN interfaces.

Program summary

Program title: VOFI

Catalogue identifier: AEYT_v1_0

Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEYT_v1_0.html

Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland

Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html

No. of lines in distributed program, including test data, etc.: 94963

No. of bytes in distributed program, including test data, etc.: 1679223

Distribution format: tar.gz

Programming language: C, with C++ and FORTRAN interfaces.

Computer: Any computer with a C compiler.

Operating system: Tested on x86 with Linux (openSUSE 13.1, Ubuntu 12.04) and Mac OS X.

Has the code been vectorized or parallelized?: The code does not need any change to be used in parallel with domain decomposition, as done for example in the Paris-Simulator code, http://parissimulator.sf.net, that is massively parallel and uses the Vofi library.

Word size: 64 bits

Classification: 4.11.

Nature of problem: The library computes the volume fraction of a cubic grid cell cut by an interface described by an implicit function.

Solution method: The library computes the integration limits along two coordinate directions and the local height function, that is the integrand of a double Gauss–Legendre integration with a variable number of nodes.

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author. Tel.: +39 051 2087720; fax: +39 051 2087747.
E-mail address: ruben.scardovelli@unibo.it (R. Scardovelli).

http://dx.doi.org/10.1016/j.cpc.2015.10.026

Restrictions: Cartesian grids with cubic cells.

Running time: Fractions of a second for a grid cell cut by the interface.

1. Introduction

Free-surface and two-phase flows are very common in nature and also in many physics, engineering, chemistry applications. The Volume-of-Fluid (VOF) method is one of several numerical techniques often used to follow the dynamical evolution of interfaces within the framework of direct numerical simulation (DNS) [1,2]. The method is based on the characteristic function $\chi(\mathbf{x}, t)$, a multidimensional Heaviside step function, with value 1 in the reference phase and 0 elsewhere, at a given time t.

The volume fraction function C is the discrete version of χ and it is given by the volume (area in two dimensions) occupied by the reference phase divided by the cell volume, hence $0 \leq C_i(t) \leq 1$, where i is the grid cell index. From the knowledge of the discrete scalar field C it is possible to reconstruct the interface, for example by numerically computing the local outward normal $\mathbf{n}$ as in the Piecewise Linear Interface Calculation (PLIC) method [3], and to estimate the local mean curvature.

The initialization of the volume fraction is trivial in empty cells, $C = 0$, or full cells, $C = 1$, of the computational mesh, while in the cells that are cut by the interface it is necessary to calculate the volume defined by the interface itself and the cell boundary.

A simple, but rather inefficient, initialization of the volume fraction in each cut cell can be achieved by considering an arbitrary number of internal points, either on a regular submesh or randomly distributed. The initial value of C is then given by the ratio of the number of points inside the reference phase with their total number. More advanced techniques for multidimensional integration are based on Monte Carlo methods with different sampling strategies to improve the efficiency of the method [4,5].

An alternative way to initialize the scalar field C in the cut cells is to use recursive local mesh refinement with a simple, linear approximation of the interface at the finest level [6]. However, this approach may require a large number of cell refinements and be very expensive in CPU time if very high accuracy is desired.

Yet another possibility is the direct integration of the analytical expression of the interface. The analytical integration becomes rather involved when the interface is a closed surface, i.e. a bubble or a droplet, which is usually described by an implicit equation. Furthermore, in each cut cell the local limits of integration along one coordinate direction are not independent of the other direction.

In this paper we consider computational grids with cubic cells and present a numerical algorithm to initialize the volume fraction field from a given implicit equation of the interface, $f(\mathbf{x}) = 0$. The algorithm locates the reference phase in the points $\mathbf{x}$ where the function $f(\mathbf{x})$ is negative [7,8].

A first assumption (hypothesis H1) of the Vofi library is that in each cell cut by the interface the implicit equation, $f(\mathbf{x}) = 0$, can be written as $x_1 = g(x_2, x_3)$, where each x_i is one of the three coordinate directions. In these cells, a numerical algorithm determines the main coordinate direction x_1 along which the values of the local height function (defined below) are explicitly computed with a root-finding routine.

The internal and external limits of integration along the other two coordinates, x_2 and x_3, are given by the intersections of the interface with the cell boundary and are computed with standard numerical tools, which include root-finding and minimum search routines.

A second assumption (hypothesis H2) of the Vofi library is that the interface cannot intersect a cell side more than twice. This assumption considerably simplifies the algorithm and it implies that characteristic lengthscales smaller than the grid spacing h_0 are usually not well resolved. At this point it is useful to note that (fact N1) interfaces are reconstructed and advected in a satisfactory way by VOF methods when their local radius of curvature and in general all relevant lengthscales are a few times the value of h_0. Thus, if fact N1 is verified, as expected when the volume fraction field is initialized, then hypotheses H1 and H2 are also verified.

The integration of the local height function determines the volume fraction C in the cell under investigation and it is performed with a double Gauss–Legendre quadrature rule with a variable number of nodes. The two-dimensional problem on a grid with square cells is also included as a special case.

We highlight the fact that even if a VOF/PLIC method can at most reconstruct exactly a linear interface, an accurate initialization of the discrete field C, as provided by the Vofi library, is nevertheless necessary for an exact initialization of the total mass. More importantly, the calculation of geometrical quantities, such as the local interface normal and mean curvature, is often based on finite differences of the volume fraction data, hence very accurate initial data are required to calculate the convergence rate with grid refinement of a given numerical scheme. Finally, we point out that VOF fields are manipulated by a broad range of algorithms. For example, while PLIC method is limited in accuracy, VOF advection schemes are not (see, e.g., [9]).

The library routines are written in `C`, and a simple software interface has been written to make the routines directly available for `FORTRAN` codes. A few two-dimensional and three-dimensional numerical tests are also provided with the software.

2. Numerical method

2.1. Cutoff function value f_h

Let $D \subset R^n$ be a computational domain and Ω the domain occupied by the reference phase defined by

$$\Omega = \{\mathbf{x} \in D : f(\mathbf{x}) \leq 0\}, \tag{1}$$

where the zero level set of $f(\mathbf{x})$ is the interface Γ and level sets with negative values occupy the interior of Ω. The interface is a plane curve when $n = 2$, and a surface when $n = 3$. Three two-dimensional interfaces are shown in Fig. 1, each of them with two level sets corresponding to $f(\mathbf{x}) = \pm l$, with $l = 0.015$. The characteristic function $\chi(\mathbf{x}, t_0)$ at the initial time t_0 is equal to 1 inside Ω and 0 elsewhere.

We then consider a Cartesian subdivision of the domain D with square/cubic cells of side h_0 and area/volume $V_0 = h_0^n$. The volume fraction $C_i(t_0)$ is then defined by the integral

$$C_i(t_0) = \frac{1}{V_0} \int_{V_i} \chi(\mathbf{x}, t_0)\, dV, \tag{2}$$

where V_i is the domain of integration of the ith grid cell. The integration is straightforward in empty or full cells, but in the cell

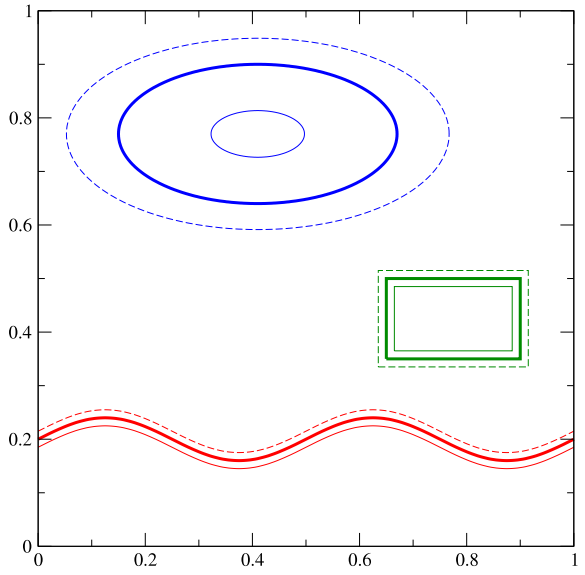

Fig. 1. Three two-dimensional interfaces (thick solid lines) described by the zero level set of a different function $f(x, y)$. For each interface the two level sets with value $l = \pm 0.015$ are also shown (thin solid lines correspond to the negative value).

cut by the interface it requires the computation of the volume defined by the cell boundary and the interface itself, where the function $f(\mathbf{x})$ is negative.

The first step of the initialization of the volume fraction field is the computation of a cutoff function value f_h which is used to determine if a grid cell is either full or empty, or if it requires a more detailed analysis. A simple search algorithm is used to determine a point $\mathbf{x}_I$ on the interface starting from an initial position $\mathbf{x}_0$. The search algorithm computes the function value $f(\mathbf{x}_0)$ and its local gradient $\nabla f(\mathbf{x}_0)$ with centered finite differences. If the function value is negative the algorithm moves up the gradient, otherwise in the opposite direction, to a new point where the values of the function and of its gradient are updated. This is the first step of an iterative procedure that stops when the interface is crossed. On the segment connecting the last two points of the search, where the function f changes sign, a root-finding algorithm is used to compute the intersection point $\mathbf{x}_I$ with the interface, since this point belongs to the zero level set of $f(\mathbf{x})$.

The cutoff value f_h is then defined by the expression

$$f_h = \max\big(|f(\mathbf{x}_I + \rho\, h_0\, \mathbf{n})|, |f(\mathbf{x}_I - \rho\, h_0\, \mathbf{n})|\big), \tag{3}$$

where $\mathbf{n}$ is the interface unit normal computed with centered finite differences, and $\rho = \sqrt{2}/4$ is the normalized maximum distance of a point on the cell boundary from the points where the function f will be evaluated when $n = 3$ ($\rho = 1/4$ when $n = 2$).

From Fig. 1 it is evident that the local spacing of the level sets changes mainly with the function $f(\mathbf{x})$, and more moderately with the level set value and along the level set itself. Clearly, the larger the spacing the smaller the cutoff value. Therefore, the cutoff value f_h has to be computed only once for an interface described by the equation $f(\mathbf{x}) = 0$, unless the local spacing of the level sets on both sides of the interface is known to vary considerably with the intersection point $\mathbf{x}_I$. However, a big increase of the cutoff value f_h with respect to the value given by Eq. (3) is usually associated with an increase of the number of cells that are analyzed in great detail but will end up being either full or empty. On the other hand, with a much smaller value of f_h tiny intersections of the interface with a cell boundary may not be detected. In the numerical tests provided with the software the cutoff value is always computed once for each interface.

The VOFI library can also be used with octree adaptive mesh refinement, by replacing the two corresponding scalar values h_0 and f_h with two one-dimensional arrays, the first one with the different grid spacings and the other one with the corresponding cutoff values.

2.2. Cell analysis and parameter $\Delta\alpha$

In each cell of the computational grid the function f is evaluated at $N_t = 3^n$ points $\mathbf{x}_j$ of a local submesh with step size $h_0/2$ along each coordinate direction, hence $N_t = 9$ in two dimensions and $N_t = 27$ in three dimensions. If $|f(\mathbf{x}_j)| = |f_j| > f_h$ the point is discarded in the cell analysis. Therefore, if all the values f_j have the same sign and satisfy $|f_j| > f_h$, then the cell is full if $f_j < 0$, and $C = 1$, otherwise it is empty and $C = 0$.

In all the other cases we need to extract more information on the local function behavior by computing its gradient with centered finite differences in the N_p points of the local submesh where $|f_j| \le f_h$, with $1 \le N_p \le N_t$. The cell-averaged value of the partial derivative in the direction w is obtained as

$$\left(\frac{\partial f}{\partial w}\right)_{ave} = \frac{1}{N_p}\sum_{j=1}^{N_p}\left(\frac{\partial f}{\partial w}\right)_j, \tag{4}$$

where $w = \{x, y, z\}$, when $n = 3$. The cell-averaged gradient component with the greatest absolute value determines the main coordinate direction x_1, while the second and third directions, x_2 and x_3 respectively, are associated with the next largest and smallest components.

The first major assumption of the VOFI library is that the interface equation, $f(\mathbf{x}) = 0$, in the cell under investigation can be written as $x_1 = g(x_2, x_3)$, hence a single-valued relation between the two local independent coordinates x_2, x_3 and the dependent main coordinate x_1. Values of the local height function, $\Delta x_1 = h(x_2, x_3)$, can be calculated by a root-finding algorithm that computes the intersection between a segment along the main direction and the interface, and will be used in the numerical integration scheme. Along the second and third directions the internal and external limits of integration are calculated with the same root-finding algorithm combined with a minimum search routine on a segment and on a cell face. In two dimensions the problem is obviously much simpler: the local height function becomes $\Delta x_1 = h(x_2)$ and only the internal limits of integration are computed.

To determine a tentative number of nodes n_* involved in the integration along the second direction x_2, we define the quantity α

$$\alpha = \left(\frac{\left(\partial f/\partial x_2\right)\,|\partial f/\partial x_2|}{\left(\partial f/\partial x_1\right)^2 + \left(\partial f/\partial x_2\right)^2}\right), \tag{5}$$

that measures the function variation along x_2 with respect to the total variation along the main and second directions. Values of α are computed at the N_p points of the local submesh in order to calculate the parameter $\Delta\alpha$ defined by the difference between the maximum and minimum values of α in the cell, $\Delta\alpha = (\alpha_{\max} - \alpha_{\min})$. The value of n_* is then determined by this parameter and the data of Table 1, ranging from $n_* = 4$, when $\Delta\alpha \le 0.1$, to $n_* = 20$, when $\Delta\alpha > 0.9$.

2.3. Internal and external limits of integration

The algorithm locates the reference phase in the region where the function $f(\mathbf{x})$ is negative. To calculate numerically the volume fraction value C in a cut cell we need to compute at different positions the local height function $h(x_2, x_3)$, that satisfies the relation $0 \le h \le h_0$. As shown in Fig. 2, to construct the h function we consider a point in terms of local coordinates (x_2, x_3), with

Table 1
Number of nodes n_* as a function of the parameter $\Delta\alpha$ defined in Eq. (5).

$\Delta\alpha$	n_*
0.00–0.10	4
0.10–0.35	8
0.35–0.65	12
0.65–0.90	16
0.90–1.00	20

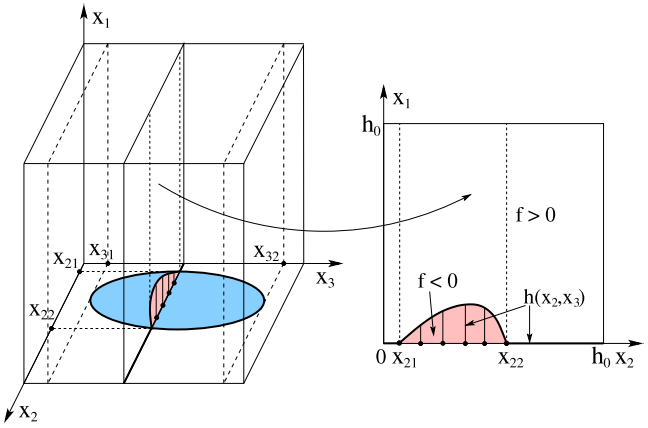

Fig. 2. The cubic cell is subdivided into three right hexahedra after the computation of the two external limits of integration x_{31} and x_{32}, with the interface being present only in the central one (left); the internal integration requires the computation of the two internal limits of integration x_{21} and x_{22} and of the local height function h (right).

$0 \le x_2, x_3 \le h_0$, and examine the segment of length h_0 along the main direction x_1. The value of the height function h at point (x_2, x_3) is the length of the fraction of the segment where the function $f(\mathbf{x})$ is negative. With this definition the function h can be extended with continuity to the points (x_2, x_3) where the segment along x_1 is not crossed by the interface.

However, even if the height function h is continuous, its first partial derivatives are in general not continuous in the points where the interface crosses the cell boundary, as seen on the right of Fig. 2. In a numerical scheme most of the integration error will be located near these points, where we approximate a function having discontinuous first derivatives with a high-order polynomial. To minimize this error we compute the intersection points of the interface with the cube sides along the third direction x_3 and subdivide the cube into rectangular hexahedra. The integration is then performed independently in each hexahedron. Similarly, for two-dimensional integration we compute the interface intersections with the sides along the x_2 direction and subdivide the square of side h_0 into rectangles.

A single interface intersection with a cell side is easily computed with the root-finding routine, while two intersections require first a minimum search on the side to detect a function sign change and then the computation of the two zeros, as required on the right of Fig. 2.

The second major assumption of the Vofi library is that three or more intersections along the same side are not considered by the algorithm, since they would imply an oscillation in the interface with a characteristic lengthscale smaller than the grid spacing h_0. In fact, this feature is not consistent with the VOF requirement that the local radius of curvature of the interface should be at least a few times larger than the side length h_0. In a three-dimensional space the interface can also intersect the cell face on a closed line, as shown on the left of Fig. 2; in this case we have to determine the two external limits x_{31} and x_{32}, relative to points that are inside the face. The minimum search routine on a cell face is first used to detect a function change of sign, then the root-finding routine is called several times to get a convergent sequence of approximations to the two external limits x_{31} and x_{32}.

Table 2
Number of nodes n_i for the internal integration, as a function of the parameter n_* of Table 1 and the non-dimensional length δ_i; number of nodes n_e for the external integration, as a function of the non-dimensional length δ_e.

δ_i	n_i	δ_e	n_e
0.0–0.1	$\min(n_*,\ 4)$	–	–
0.1–0.2	$\min(n_*,\ 8)$	0.0–0.1	8
0.2–0.4	$\min(n_*,\ 12)$	0.1–0.3	12
0.3–0.6	$\min(n_*,\ 16)$	0.3–0.5	16
0.6–1.0	$\min(n_*,\ 20)$	0.5–1.0	20

The root-finding routine is based on a hybrid algorithm that combines the bisection and secant methods [10], the minimum search on a line is based on Brent's method [11], while a preconditioned conjugate gradient method, with the Polak–Ribiere's expression for the parameter to update the conjugate direction [12,13], is used for the minimum search on a cell face. A more detailed analysis of the basic numerical tools that have been implemented in the Vofi library can be found in [8].

2.4. Numerical integration

In three dimensions the cubic cell is subdivided into rectangular hexahedra and a double Gauss–Legendre integration is done in each hexahedron cut by the interface. In the case shown in Fig. 2 the cell is subdivided into three hexahedra, with the interface being present only in the middle one. Therefore, the algorithm computes the volume fraction with the following numerical integration along the third direction x_3

$$C(t_0) = \frac{1}{V_0}\int_V \chi(\mathbf{x}, t_0)\, dV = \frac{1}{h_0}\int_{x_{31}}^{x_{32}} A(x_3)\, dx_3 \approx \frac{x_{32}-x_{31}}{2\,h_0}\sum_{k=1}^{n_e}\omega_k\, A(x_{3k}), \tag{6}$$

where the coefficients ω_k are the integration weights, V_0 is the cell volume and $A(x_{3k})$ the normalized area that comes from the integration of the local height function $h(x_2, x_{3k})$ along the second direction x_2

$$A(x_{3k}) = \frac{1}{h_0^2}\int_{x_{21}(x_{3k})}^{x_{22}(x_{3k})} h(x_2, x_{3k})\, dx_2 \approx \frac{x_{22}(x_{3k})-x_{21}(x_{3k})}{2\,h_0^2}\sum_{j=1}^{n_i}\omega_j\, h(x_{2j}, x_{3k}). \tag{7}$$

To determine the number of nodes n_i and n_e we consider the two non-dimensional lengths $\delta_i = (x_{22}(x_{3k}) - x_{21}(x_{3k}))/h_0$ and $\delta_e = (x_{32} - x_{31})/h_0$. The number of nodes n_i of the internal integration is determined from Table 2 as a function of the length δ_i and the previously-defined parameter n_*, while the number of nodes n_e of the external integration is only a function of the length δ_e. In the present version of the software we consider only 5 values for the number of nodes, that are given in Table 1. The position x_k and the integration weight w_k of each node are stored as internal parameters.

To illustrate the entire procedure we consider the ellipsoid $(x'/4)^2 + (y'/5)^2 + (z'/6)^2 = 1$. The axis z' is parallel to the coordinate axis z, while the other two axis x' and y' are rotated 60 degrees counterclockwise with respect to x and y. The ellipsoid center is at $(0.35, 0.35, -5.97)$. The intersection of the ellipsoid with the plane $z = 0$ is the ellipse shown in Fig. 3, while the ellipsoidal cap intersects three grid cells of side $h_0 = 1$. In each

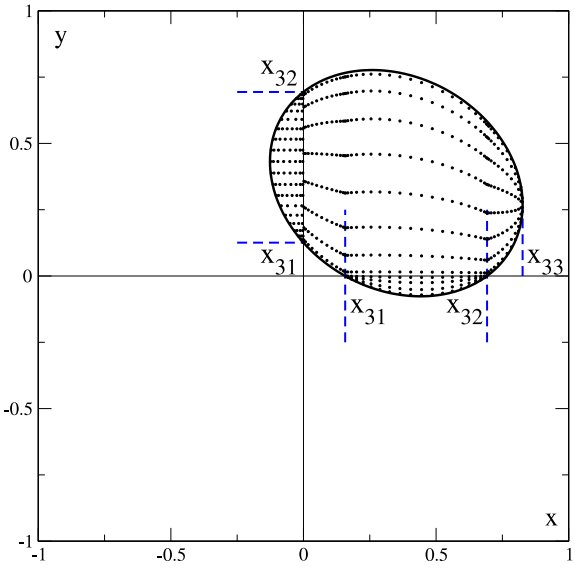

Fig. 3. Subdivision of a 2D projection of an ellipsoidal cap intersecting three grid cells of side $h_0 = 1$. The primary direction is always found to be $x_1 = z$, while for the top-left cell $x_3 = y$, and for the other two cut cells $x_3 = x$. The coordinates x_{3k} define the thickness of the right hexahedra. The full circles represent the position in the plane $z = 0$ of the points where the local height function h is calculated for the numerical integration.

cut cell the numerical algorithm determines that the main direction x_1 is along the z coordinate, while for the top-left cell the third direction x_3 is along the y axis and the cell is subdivided into three hexahedra; for the other two cut cells the subdivision is along the x axis, with three and four hexahedra respectively. Fig. 3 also shows the position of the nodes in the plane $z = 0$ where the local height function is calculated. The volume V_a of the ellipsoidal cap is easily computed analytically, while its numerical approximation V_n is given by the sum of the volume fraction of the three cut cells of Fig. 3. The error of the numerical integration with the VOFI library is $|V_a - V_n| \approx 2.615\,10^{-10}$, while the relative error is $E_1 = |V_a - V_n|/V_a \approx 2.779\,10^{-8}$.

3. User manual

3.1. *FORTRAN* calls

The VOFI library has been developed in C and for its usage in FORTRAN codes we have written two straightforward interface routines that provide the usual conversion from "pointer to variable type" to "variable type" and call the corresponding C function. Both functions return a real number (in double precision). The first function, `vofi_get_fh`, computes the cutoff value f_h and usually is called only once for each implicit function

```
fh = vofi_get_fh(impl_func,x0,h0,ndim0,ix0)
```

where the input arguments are defined as follows

- `impl_func`: the external function that computes $f(\mathbf{x})$ and is expected to be declared as

```
function impl_func(x)
real(8) :: impl_func
real(8), dimension(3) :: x
...
end function impl_func
```

- `x0`: initial position $\mathbf{x}_0$ (1D array of real numbers)
- `h0`: grid spacing (real number)
- `ndim0`: space dimension (integer number, either 2 or 3)
- `ix0`: switch either to use or not to use the default value for $\mathbf{x}_0$ (integer number, respectively 0 or 1); the default value is $\mathbf{x}_0 = (0.5, 0.5, 0.5)$.

The second function, `vofi_get_cc`, computes the volume fraction in a given cell

```
cc = vofi_get_cc(impl_func,x0,h0,fh,ndim0)
```

where now

- `x0`: cell vertex position with smallest coordinates (1D array of real numbers)

3.2. *C/C++ calls*

The relevant declarations in C are

```
typedef double (*integrand) (const double []);
double vofi_Get_fh(integrand,const double [],const
double, const int, const int);
double vofi_Get_cc(integrand,const double [],const
double, const double, const int);
```

These prototypes are contained in the header file `vofi.h` that should be included when using the VOFI routines. The function call in C is similar to the FORTRAN call, for example for the cutoff function value f_h

```
fh = vofi_Get_fh(impl_func,x0,h0,ndim0,ix0);
```

while the user-provided implicit function should begin as

```
double impl_func(const double x[])
  {...}
```

To link the library it is necessary to add `-lvofi` to the compiler command line. More details are given in the README file of the software distribution.

4. Tests and comparisons

4.1. Implicit representation of the interface

Three two-dimensional interfaces are shown in Fig. 1. Each interface is given by the zero level set of a different function $f(x, y)$, and the figure also shows the level sets corresponding to the two values $l = \pm 0.015$. The ellipse and the rectangle are closed curves, while the sinusoidal wave is an open interface. For the wave the implicit function is

$$f(x, y) = y - \big(y_0 + a_0 \sin(\beta_0 x + \gamma_0)\big), \tag{8}$$

with $(y_0, a_0, \beta_0, \gamma_0) = (0.2, 0.04, 4\pi, 0)$. The reference phase is below the interface line, where the function $f(x, y)$ is negative. For the two closed interfaces, the reference phase is positioned inside the geometric figure when

$$f(x, y) = \frac{1}{c_0^2}(x - x_0)^2 + (y - y_0)^2 - r_0^2, \tag{9}$$

for the ellipse, with $(x_0, y_0, c_0, r_0) = (0.41, 0.77, 2, 0.13)$, and

$$f(x, y) = \max\big(f_1, f_2, f_3, f_4\big), \tag{10}$$

for the rectangle, where

$$f_1 = x_1 - x, \quad f_2 = x - x_2, \quad f_3 = y_1 - y, \quad f_4 = y - y_2, \tag{11}$$

and $(x_1, x_2, y_1, y_2) = (0.65, 0.9, 0.35, 0.5)$. These functions are easily extended to the three-dimensional space.

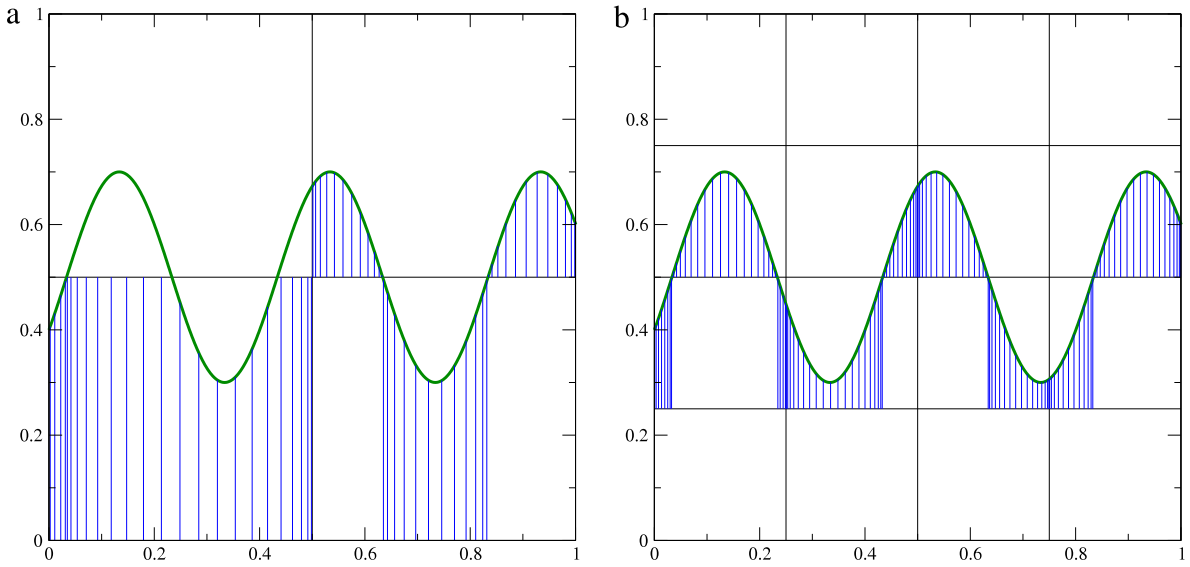

Fig. 4. Initialization of the volume fraction field for a sinusoidal wave in the unit square with m^2 cells: (a) $m = 2$; (b) $m = 4$.

4.2. Two-dimensional tests

For an accurate reconstruction and advection of a circular interface with a PLIC algorithm, the ratio between the circle's radius of curvature r_0 and the grid spacing h_0 should satisfy $r_0/h_0 \geq 4$. For this reason, the VOFI library has been designed to resolve up to two intersections between the interface and a cell side, in order to determine the integration limits, and one intersection on a segment along the main direction x_1, to compute the local height function $h(x_2, x_3)$. However, it is reasonable to check how the library behaves when these requirements are not satisfied.

For the sinusoidal wave

$$f(x, y) = y - \left(\frac{1}{2} + \frac{1}{5}\,\sin\left(5\,\pi\,x - \frac{\pi}{6}\right)\right), \tag{12}$$

the minimum radius of curvature is $r_{\min} = (5\,\pi^2)^{-1}$ and the interface is clearly underresolved for both grids of Fig. 4. At the lowest resolution the interface intersects a horizontal cell side three times, as seen in the left column of Fig. 4a. The root-finding routine converges to the left root and both cells in the column are subdivided into two rectangles. A function sign check is always performed in the central height of each rectangle. In the bottom-left cell a sign change is detected in both rectangles and the numerical integration is performed with 4 and 20 nodes, respectively. In the top-left cell no sign change is detected, and since the function is positive the volume fraction of this cell is equal to zero. On the right column the VOFI library computes the two interface intersections with a horizontal cell side and the two cells are then subdivided into three rectangles. All rectangles are correctly selected as full, empty or cut by the interface. In Fig. 4b the number of cells is doubled along each coordinate direction and all the cells are correctly subdivided even if the ratio $r_{\min}/h_0 = 4/(5\,\pi^2) < 1$.

For the circle

$$f(x, y) = (x - 0.623)^2 + (y - 0.377)^2 - (0.25)^2, \tag{13}$$

inside a unit square with the 4 grid resolutions shown in Fig. 5, the ratio between the circle's radius and the grid spacing varies from 0.5 to 2.5. The relative error is defined by $E_1 = |A_a - A_n|/A_a$, where A_n is the numerical area computed with the VOFI library and A_a its analytical value. At the lowest resolution, with 2^2 cells, in the bottom-right cell of Fig. 5a the y-axis has been selected as main direction, one intersection with the horizontal sides is computed and the cell is subdivided into two rectangles. In the left rectangle a sign change is detected and the integration is performed with 20 nodes, while in the right rectangle no sign change is detected and since the function is positive the rectangle is assumed to be empty. As a matter of fact, there is a small range where the relation $y = y(x)$ has two values, and between these two values the function $f(x, y)$ is negative. However, the model assumes that the relation is single-valued along the main direction, so this area is not computed at all. The results of Table 3 show that the relative error is mainly due to the missing area, since it is about constant with the number of integration points.

For the next two resolutions with 4^2 and 6^2 grid cells, shown respectively in Fig. 5b and c, the results are systematically better at the lower resolution. This is due to the particular symmetrical alignment of the circle with respect to the grid lines of the lower resolution, since the circle center almost coincides with a cell center. In Fig. 5d there are five cells with a mark, three of them are empty and two are full. However, these cells are very close to the interface and for at least one point $\mathbf{x}_j$ of the local subgrid the relation $|f_j| \leq f_h$ is satisfied, hence a more detailed analysis in the cells is carried out even if they are not cut by the interface.

The results of Table 3 show that for this test the relative error with the proposed algorithm with a variable number of nodes corresponds to that with a fixed number of nodes comprised between 12 and 16. However, the average number of integration nodes in a cut cell diminishes with the grid resolution, and with 10^2 grid cells this number is $n_{ave} = 8.4$ (with 20^2 cells the average number is $n_{ave} = 7.7$).

For a comparison with other initialization schemes, we consider again the circle of Eq. (13) in the grid with 10^2 cells, and in every cut cell a Cartesian subgrid with n_0 points evenly spaced along each coordinate direction, and count the number of nodes n_n where the implicit function value is negative. We define the volume fraction as the ratio $C = n_n/n_0^2$, then with $n_0 = 10$ the relative error is $E_1 \approx 4.3\,10^{-3}$, while with $n_0 = 100$ we find $E_1 \approx 5.4\,10^{-5}$. Another strategy to initialize the scalar field C is that of a recursive local mesh refinement in the cut cells, where at the finest level the interface can be approximated by a linear interface and the cut area can be easily computed. With four levels of subdivisions an extrapolation of the results of [6] gives $E_1 \approx 10^{-4}$. For this approach, a simple algorithm based on the function sign check on the four subcell corners will require a few dozen function calls to determine the cut subcells at the highest resolution and a similar number to compute the intersections of the interface with the grid

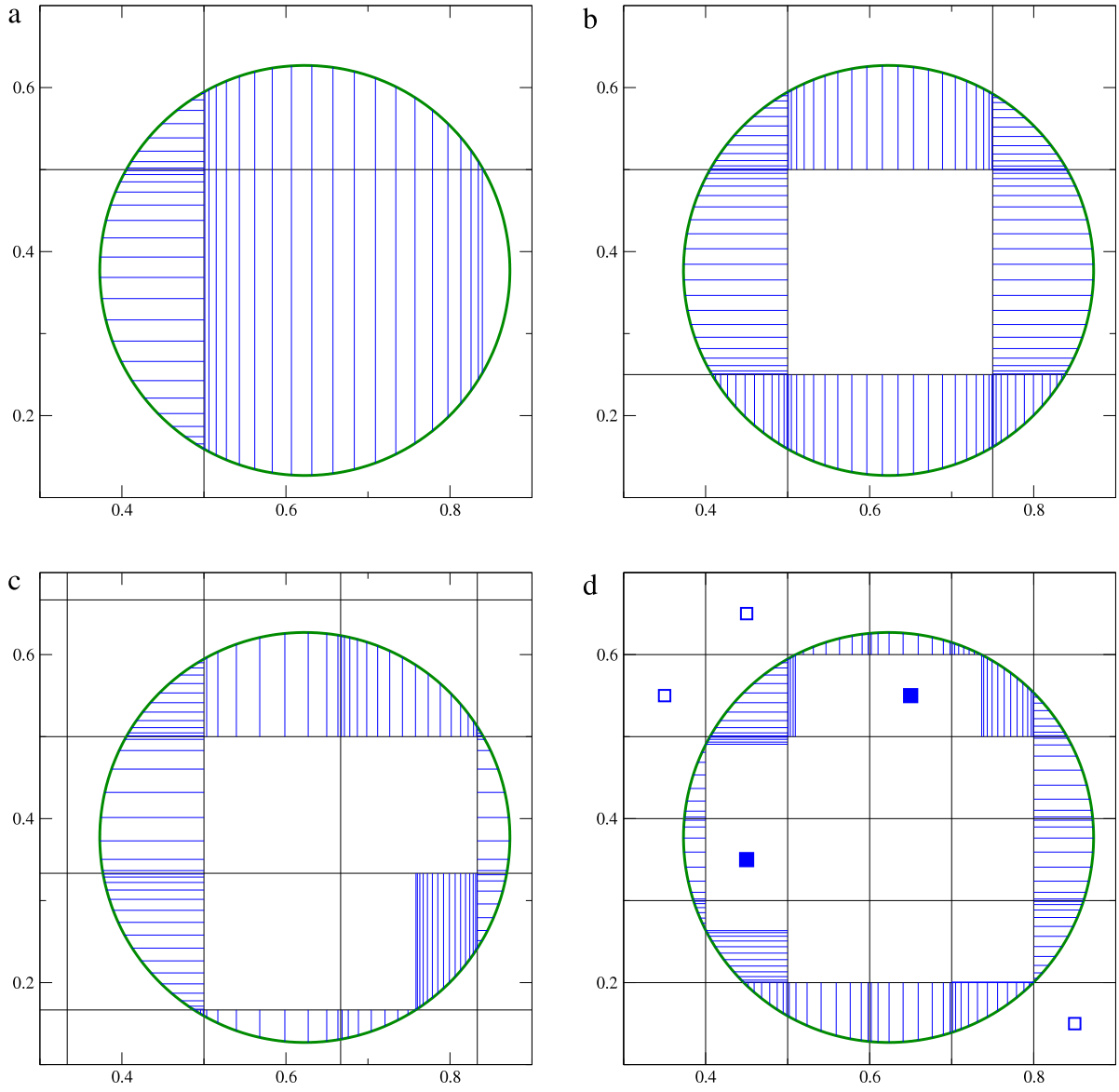

Fig. 5. Initialization of the volume fraction field for a circle in the unit square with m^2 cells: (a) $m = 2$; (b) $m = 4$; (c) $m = 6$; (d) $m = 10$.

Table 3
Total number of cells m^2 in the unit square for the initialization of the circle of Eq. (13) and number of cells m_{cut} cut by the interface, relative error E_1 with a fixed number of integration nodes: 4, 8, 12, 16, 20 respectively, relative error E_1 with the average number of nodes n_{ave} selected by the VOFI library.

m^2	m_{cut}	E_1 (4)	E_1 (8)	E_1 (12)	E_1 (16)	E_1 (20)	E_1 (n_{ave})
2^2	4	2.72e−02	2.74e−02	2.74e−02	2.74e−02	2.74e−02	2.74e−02 (17)
4^2	8	2.36e−06	3.89e−11	1.98e−15	2.83e−16	1.41e−16	1.41e−15 (16)
6^2	12	3.78e−06	6.47e−10	1.86e−13	2.83e−16	2.83e−16	8.00e−14 (9.7)
10^2	20	2.38e−07	1.03e−11	8.48e−16	1.41e−16	2.82e−16	7.07e−16 (8.4)

lines for its linear approximation, if a root-finding routine is used. Hence, we expect a total number again in the order of 10^2 calls.

The relative error with the VOFI library and four integration nodes is $E_1 \approx 2.38\ 10^{-7}$, see Table 3. The number of function calls to compute the cutoff value f_h is 17, while 9 calls are necessary to determine if a cell is either empty or full and an average of 26.2 calls for the five cells that require a more detailed analysis. To determine the volume fraction C in a cut cell an average number of about 64 calls is required, and between 5 and 6 calls for each additional integration node. Hence, for a similar number of function calls the VOFI library provides a more accurate initialization of the volume fraction scalar field.

A collection of test cases, including ellipses, rectangles, sinusoidal and Gaussian lines, is present in the software distribution.

4.3. Three-dimensional tests

For the quarter of a spherical segment shown in Fig. 6, the ratio between the radius of the sphere and the length of the cell side is $r_0/h_0 = 2$ and for this simple case we can use two different implicit functions

$$f_1(x, y, z) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r_0^2,$$
$$f_2(x, y, z) = z - z_0 - \left(r_0^2 - (x - x_0)^2 - (y - y_0)^2\right)^{1/2}. \quad (14)$$

Table 4
Number of external nodes n_e, number of calls of the two functions f_1 and f_2 of Eq. (14) and relative error $E_1 = |V_a - V_n|/V_a$ for the computation of the volume of the quarter of the spherical segment of Fig. 6 with the VOFI library; number of calls of the function f_3 of Eq. (15), up to a maximum of 100 000, and relative error E_1 with `Cuhre` and `Divonne` algorithms.

VOFI				Cuhre		Divonne	
n_e	f_1 calls	f_2 calls	E_1	$f_3$calls	E_1	$f_3$calls	E_1
4	44	305	2.78e−04	325	2.11e−04	2148	1.53e−04
8	739	466	9.90e−06	4 355	1.09e−05	23556	1.29e−05
12	1028	623	1.40e−06	20 865	1.40e−06	48231	1.61e−06
16	1318	781	3.45e−07	37 635	3.62e−07	***	***
20	1585	930	1.16e−07	65 715	1.19e−07	***	***

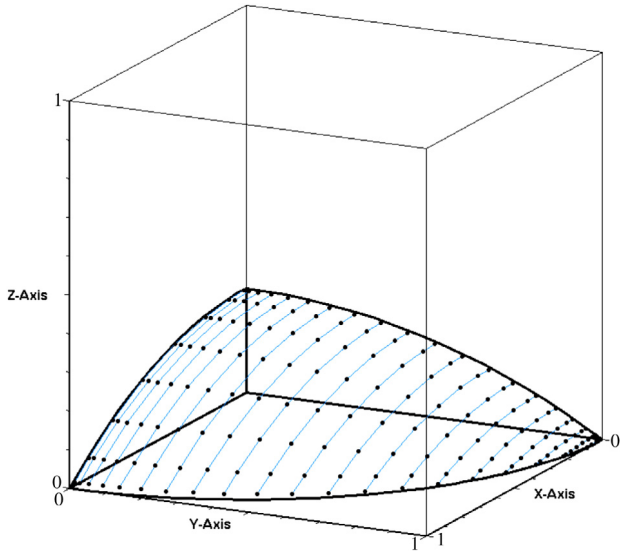

Fig. 6. Top of the height segments for the initialization of the volume fraction function with the reference phase inside a quarter of a spherical segment; the sphere center is at $(0, 0, -\sqrt{3})$ with radius $r_0 = 2$, the cell side is $h_0 = 1$.

For both functions the primary direction is $x_1 = z$ while $x_2 = x$ and $x_3 = y$, but the different formulation has a strong impact on the number of iterations that are required to compute the local height function. The function f_1 is quadratic in z and an average number of 7 function calls is required to compute the local height h: 2 calls for the function value at the end points of the initial segment and 5 iterations to converge; the function f_2 is instead linear in z and only 3 calls are required: 2 for the end points and only one iteration.

The computation of the cutoff function value f_h requires about 30 function calls, when the initial point is the default value $\mathbf{x}_0 = (0.5, 0.5, 0.5)$, while 105 more calls are necessary to determine the main, second and third directions. Finally, as shown in Fig. 6, there is only one hexahedron and to determine the external limits of integration the function needs to be called about 15 times. Hence, about 150 calls are required for the set-up of the problem, independently of the number of external and internal nodes.

For the internal integrations Table 2 gives us $n_i = 8$, except when $n_e = 20$ and for the rightmost line of Fig. 6 where we have $n_i = 4$. An average number of 16 function calls are required to determine the limits of each internal integration. For this test the internal integrations of Eq. (7) are very accurate with an error $E_{1i} \approx 10^{-14}$. On the other hand the external integration of Eq. (6) converges much more slowly. Therefore, we consider a sequence of different values of n_e, varying from 4 to 20, and compute the corresponding integration error, which is given in Table 4.

For a comparison with an open-source software, we have considered the CUBA library for multidimensional numerical integration with deterministic and Monte Carlo methods [5]. The CUBA library has been integrated in the Maple software, and it has been also compared with Mathematica's `NIntegrate` function [5]. In principle, we could use the value of the relative error from the VOFI library in Table 4 and use it as the prescribed accuracy for the numerical integration with the CUBA routines. However, we observe that the actual relative error it is always much smaller than the prescribed accuracy, therefore to determine the number of function calls we slowly decrease the prescribed accuracy until the fluctuating relative error is not bigger than the value from the VOFI library. `Cuhre` is a deterministic algorithm that recursively bisects the integration domain into smaller subdomains and applies the same integration rule of polynomial degree to each of them. `Divonne` is a Monte Carlo algorithm that we have found for our simple example to be significantly faster than the other two algorithms, `Vegas` and `Suave`, that are present in the CUBA library. For these algorithms, the integrand must be an explicit function which is easily derived from the function f_2 of Eq. (14)

$$h = f_3(x, y) = \max\left(0, z_0 + \left(r_0^2 - (x - x_0)^2 - (y - y_0)^2\right)^{1/2}\right). \tag{15}$$

Since the function f_3 is now explicit, no iteration is required to compute the local height h. In Table 4 we present the results obtained with the two routines from the CUBA library as well. In the comparison the reader should remember that VOFI requires about 135 function calls to compute the cutoff function value f_h and to determine the main, second and third directions.

A collection of three-dimensional test cases, including the ellipsoidal cap of Fig. 3, spheres and trigonometric surfaces, is present in the software distribution.

5. Conclusions

The VOFI library is a software package written in `C` aimed at computing the volume fraction scalar field from an implicit analytical representation of the interface. In the cells cut by the interface the algorithm first calculates the main, second and third coordinate directions. The internal and external limits of integration are then determined by the interface intersections with the cell boundary along the second and third directions, while the local heights are computed along the main direction. The numerical integration of the height function in three dimensions is performed with a double Gauss–Legendre integration with a variable number of nodes. The tests presented in the study show very accurate results in both two and three dimensions. The source code has been compiled with the GNU `C` compiler, but it should compile with any standard ANSI `C` compiler. The `C` functions can be called directly from `FORTRAN` using a few interface routines.

References

[1] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annu. Rev. Fluid Mech. 31 (1999) 567–603.
[2] G. Tryggvason, R. Scardovelli, S. Zaleski, Direct Numerical Simulations of Gas–Liquid Multiphase Flows, Cambridge University Press, Cambridge, UK, 2011.
[3] W.J. Rider, D.B. Kothe, Reconstructing volume tracking, J. Comput. Phys. 141 (1998) 112–152.

[4] M. Evans, T. Swartz, Approximating Integrals via Monte Carlo and Deterministic Methods, Oxford University Press, New York, USA, 2000.
[5] T. Hahn, Cuba—A library for multidimensional numerical integration, Comput. Phys. Comm. 168 (2005) 78–95.
[6] S.J. Cummins, M.M. Francois, D.B. Kothe, Estimating curvature from volume fractions, Comput. Struct. 83 (2005) 425–434.
[7] G. Bornia, A. Cervone, S. Manservisi, R. Scardovelli, S. Zaleski, On the properties and limitations of the height function method in two-dimensional Cartesian geometry, J. Comput. Phys. 230 (2011) 851–862.
[8] S. Bnà, S. Manservisi, R. Scardovelli, P. Yecko, S. Zaleski, Numerical integration of implicit functions for the initialization of the VOF function, Comput. Fluids 113 (2015) 42–52.
[9] E. Aulisa, S. Manservisi, R. Scardovelli, S. Zaleski, A geometrical area-preserving volume-of-fluid advection method, J. Comput. Phys. 192 (2003) 355–364.
[10] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Numerical Recipes in C: The Art of Scientific Computing, second ed., Cambridge University Press, New York, USA, 1993.
[11] R.P. Brent, Algorithms for Minimization Without Derivatives, Prentice–Hall, Englewood Cliffs, NJ, USA, 1973.
[12] J.R. Shewchuk, An Iintroduction to the Conjugate Gradient Method Without the Agonizing Pain. Technical Report, Carnegie Mellon University, Pittsburgh, PA, USA, 1994. Tech. Rep. CMU-CS-94-125.
[13] E. Polak, G. Ribiere, Note sur la convergence de méthodes de directions conjuguées, ESAIM Math. Model. Numer. Anal 3 (1969) 35–43.