Finite element approximation of a moving boundary problem with curvature dependent speed

Cornel Marius Murea
Laboratoire de Mathématiques et Applications,
Université de Haute-Alsace,
4, rue des Frères Lumière, 68093 MULHOUSE Cedex, France,
e-mail: c.murea@uha.fr

George Hentschel
Emory University, Department of Physics,
Atlanta, GA 30322 USA,
e-mail: phshgeh@physics.emory.edu

Key words. Finite Element Method, single phase Hele-Shaw problem with surface tension

1 Introduction

We study a moving boundary problem similar to so called single phase Hele-Shaw problem with surface tension. We have to determine the evolution of a two dimensional moving domain. The velocity of a part of the boundary is parallel to the gradient of a pressure. The pressure is the solution of a Poisson problem with Dirichlet boundary conditions depending on the curvature of the boundary. This model describes the growth of some biological organisms. In a classical Hele-Shaw problem, the boundary moves with a normal velocity.

We present two numerical algorithms in order to approximate the problem with non-normal velocity of the boundary.

In the first algorithm, the boundary of the domain is approached by a polygon and the pressure is computed by the Finite Element Method. The computed pressure is a piecewise linear function, globally continuous. The curvature is computed as the inverse of the ray of the circle passing through three consecutive vertices of the boundary.

We compute the gradient of the pressure in a vertex of the mesh, as a weighted mean of the gradients in the neighborhood triangles.

For the time discretization, we use the forward finite difference Euler's scheme.

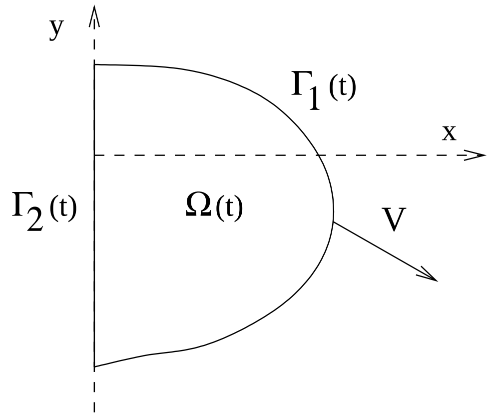
A dynamic mesh technique is used in order to generate a triangular mesh at each time step.

In the second algorithm, the boundary is approached by cubic spline interpolation which gives a curve twice continuously differentiable. The curvature is computed using the parametrization of the splines.

At each time step, a new mesh is generated.

2 Presentation of the problem

We study the evolution of a bounded connected open domain $\Omega(t)$ of $\mathbb{R}^2$ with boundary $\partial\Omega(t) = \Gamma_1(t) \cup \Gamma_2(t)$, where $\Gamma_1(t)$ and $\Gamma_2(t)$ are two non empty subsets of $\partial\Omega(t)$. Here $t \geq 0$ is the time. We assume that $\Gamma_1(t)$ is a non closed curve of class $\mathcal{C}^2$ and its ends evolve on the Oy ax. The boundary $\Gamma_2(t)$ is the segment which has the same ends as $\Gamma_1(t)$. Let ν denote the outer unit normal vector to the boundary.



In the moving domain $\Omega(t)$, we have to find the pressure $p(\cdot, t) : \Omega(t) \to \mathbb{R}$, such that

$$\begin{cases} -\Delta p &= S & \text{in } \Omega(t) \\ p &= \sigma \cdot \kappa & \text{on } \Gamma_1(t) \\ \dfrac{\partial p}{\partial \nu} &= 0 & \text{on } \Gamma_2(t) \end{cases} \tag{1}$$

where S, σ are real constants and κ is the curvature of $\Gamma_1(t)$.

The boundary $\Gamma_1(t)$ evolves according to the law

$$V = -\alpha \nabla p \tag{2}$$

where V is the velocity of $\Gamma_1(t)$ and α is a positive constant.

We know the initial domain

$$\Omega(0) = \Omega^0. \tag{3}$$

We consider the problem of determining the evolution of $\Omega(t)$ and to find the pressure $p(\cdot, t)$ for $t \in [0, T]$, where $T > 0$ is a given real constant.

It is convenient to describe the curve $\Gamma_1(t)$ by the parametric coordinates

$$\begin{array}{rcl} x & = & r_1(\theta, t), \\ y & = & r_2(\theta, t), \end{array} \qquad \theta \in [a, b].$$

Let us introduce the following generalized cylinder:

$$\Omega_T = \bigcup_{t \in]0,T[} (\Omega(t) \times \{t\}).$$

We can rewrite the problem (1), (2) and (3) as following:
find $r = (r_1, r_2) : [a, b] \times [0, T] \to \mathbb{R}^2$ and $p : \overline{\Omega}_T \to \mathbb{R}$, such that

$$\begin{array}{rcll} \dfrac{\partial r_1}{\partial t}(\theta, t) & = & -\alpha \dfrac{\partial p}{\partial x}(r_1(\theta, t), r_2(\theta, t), t), & \forall \theta \in [a, b], \forall t \in]0, T[\\ \dfrac{\partial r_2}{\partial t}(\theta, t) & = & -\alpha \dfrac{\partial p}{\partial y}(r_1(\theta, t), r_2(\theta, t), t), & \forall \theta \in [a, b], \forall t \in]0, T[\\ r(\theta, 0) & = & (r_1^0(\theta), r_2^0(\theta)), & \forall \theta \in [a, b] \end{array}$$

where $r^0 = (r_1^0, r_2^0)$ is a parametric representation of $\Gamma_1(0)$ and $p(x, y, t)$ is the solution of (1).

Since $\frac{\partial p}{\partial \nu} = 0$ on $\Gamma_2(t)$, we have

$$\begin{array}{l} \frac{\partial p}{\partial x}(r_1(a, t), r_2(a, t), t) = 0, \forall t \in]0, T[\\ \frac{\partial p}{\partial x}(r_1(b, t), r_2(b, t), t) = 0, \forall t \in]0, T[, \end{array}$$

then $r_1(a, t) = r_1(b, t) = 0$, $\forall t \in]0, T[$.

The boundary could be parametrized in multiple ways, but the solution must be independent of parametrization.

This problem is similar to so called single phase Hele-Shaw problem with surface tension, but in our case the speed of the boundary isn't normal to the boundary. As it was shown in [4], the shape of the moving domain is determined solely by its normal velocity.

The existence and uniqueness of classical solution for the single phase Hele-Shaw problem was proved in [2].

3 Numerical methods

3.1 The first algorithm

For the time discretization, we use the forward finite differences Euler's scheme. We denote by Δt the time step and by $N = T/\Delta t$ the number of time steps.

We approximate $\Gamma_1(n\Delta t)$ by a polygonal line Γ_1^n of vertices (x_i^n, y_i^n) for $i = 0, \ldots, M$. We have $x_0^n = x_M^n = 0$, for all n. We denote by Ω^n the polygonal domain bounded by Γ_1^n and the Oy ax.

For each vertex (x_i, y_i) of Γ_1^n, we compute the discrete curvature $\kappa^n(x_i, y_i)$ as the inverse of the ray of the circle passing through the three points (x_{i-1}^n, y_{i-1}^n), (x_i^n, y_i^n) and (x_{i+1}^n, y_{i+1}^n).

The problem (1) is solved numerically by the Finite Element Method. The computed pressure p^n is approached by P_1 function, globally continuous.

We follow [6] for computing the discrete gradient of p^n. Let A be a vertex of Γ_1^n. We denote by $star(A)$ the set of all triangles T of the mesh such that A is a vertex of T. We compute the discrete gradient of p^n at the point A as following:

$$\frac{\sum_{T\in star(A)} Area(T) \cdot \nabla p^n{}_{|T}}{\sum_{T\in star(A)} Area(T)},$$

where $p^n{}_{|T}$ is the linear function representing the restriction of the function p^n on the triangle T.

Algorithm 1

Generate a triangular mesh for Ω^0 using **freefem+** [1].
for each n **from** 0 **to** $N-1$ **do**

Step 1: Compute the discrete curvature $\kappa^n(x_i, y_i)$ at each vertex (x_i, y_i) of Γ_1^n.

Step 2: Compute p^n by the Finite Element Method

$$\begin{cases} -\Delta p^n(x,y) & = & S, & \text{in } \Omega^n \\ p^n(x_i,y_i) & = & \sigma\,\kappa^n(x_i,y_i), & \forall (x_i,y_i) \text{ vertex of } \Gamma_1^n \\ \dfrac{\partial p^n}{\partial \nu}(x,y) & = & 0, & \text{on } \Gamma_2^n. \end{cases}$$

Step 3: Compute the discrete gradient of p^n at each vertex of Γ_1^n.

Step 4: Compute the vertices of Γ_1^{n+1}

$$\begin{aligned} x_i^{n+1} &= x_i^n - \alpha\Delta t \frac{\partial p^n}{\partial x}(x_i^n, y_i^n) \\ y_i^{n+1} &= y_i^n - \alpha\Delta t \frac{\partial p^n}{\partial y}(x_i^n, y_i^n) \end{aligned}$$

for each vertex $(x_i^n, y_i^n) \in \Gamma_1^n$, $i = 0, 1, \ldots, M$, then set $x_0^{n+1} = x_M^{n+1} = 0$.

Step 5: The boundary Γ_2^{n+1} is the segment of ends $\left(x_M^{n+1}, y_M^{n+1}\right)$ and $\left(x_0^{n+1}, y_0^{n+1}\right)$.

Step 6: Compute a triangular dynamic mesh for Ω^{n+1}, where $\partial\Omega^{n+1} = \Gamma_1^{n+1} \cup \Gamma_2^{n+1}$ using the algorithm [5].

endfor;

3.2 The second algorithm

Let (x_i^n, y_i^n) for $i = 0, \ldots, M$ be points on $\Gamma_1(n\Delta t)$. Let $\{a = s_0 < s_1 < \ldots < s_M = b\}$ be a partition of an interval $[a, b]$. We will compute the interpolating cubic spline functions $\Gamma_1^n = \{(x(s), y(s))\,,\ s \in [a, b]\}$ with the properties:

- $x(s)$ and $y(s)$ are twice continuously differentiable on $[a, b]$,
- $x(s)$ and $y(s)$ coincide on every subinterval $[s_i, s_{i+1}]$, $i = 0, \ldots, M-1$ with polynomials of degree three,
- $x(s_i) = x_i^n$ and $y(s_i) = y_i^n$ for $i = 0, \ldots, M$,
- $x''(a) = x''(b) = 0$ and $y'(a) = y'(b) = 0$.

For the numerical tests, we have chosen $s_i = i$, for $i = 0, \ldots, M$.
In order to prevent the oscilations, we could choose $a = 0$ and

$$s_{i+1} - s_i = \sqrt{\left(x_i^n - x_{i+1}^n\right)^2 + \left(y_i^n - y_{i+1}^n\right)^2}.$$

The curve $(x(s), y(s))$, $s \in [a, b]$ has a continuous curvature given by

$$k(s) = \frac{x'(s)y''(s) - x''(s)y'(s)}{\left((x'(s))^2 + (y'(s))^2\right)^{3/2}}. \tag{4}$$

Algorithm 2

Let (x_i^0, y_i^0) for $i = 0, \ldots, M$ be points on $\Gamma_1(0)$.
for each n **from** 0 **to** $N-1$ **do**

Step 1: Compute the cubic spline functions

$$\Gamma_1^n = \{(x(s), y(s))\,,\ s \in [a, b]\}$$

Step 2: Compute the curvature $\kappa^n(x_i, y_i)$ at each vertex (x_i^n, y_i^n) using (4).

Step 3: Generate a triangular mesh for Ω^n using **freefem+** [1], where $\partial\Omega^n = \Gamma_1^n \cup \Gamma_2^n$ and Γ_2^n is the segment of ends (x_M^n, y_M^n) and (x_0^n, y_0^n).

Step 4: Step 5: Step 6: Compute p^n, ∇p^n and (x_i^{n+1}, y_i^{n+1}) as in the Algorithm 1.

endfor;

4 Numerical tests

In the case when the initial domain is a semicircle of ray R_0 and if we set the parametric representation of $\Gamma_1(0)$ as

$$\begin{array}{rcl} r_1^0(\theta) &=& R_0\cos(\theta), \\ r_2^0(\theta) &=& R_0\sin(\theta), \end{array} \qquad \theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

the evolution of the boundary $\Gamma_1(t)$ is described by

$$\begin{array}{rcl} r_1(\theta,t) &=& R_0\, e^{\frac{\alpha St}{2}}\cos(\theta), \\ r_2(\theta,t) &=& R_0\, e^{\frac{\alpha St}{2}}\sin(\theta), \end{array} \qquad \theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \quad t > 0.$$

The pressure has the form

$$p(x,y,t) = \frac{S}{4}\left(R_0^2\, e^{\alpha St} - x^2 - y^2\right) + \frac{\sigma}{R_0\, e^{\frac{\alpha St}{2}}}.$$

The algorithm has been implemented using the programming language C++ and the Finite Element classes of F. Hecht [3]. The numerical results were displayed using `gnuplot`.

The first simulation was performed using the **Algorithm 1** for $R_0 = 1$, $S = 4$, $\sigma = 1$, $\alpha = 0.5$, $\Delta t = 0.05$. The number of time steps is $N = 10$.

A dynamic mesh technique is used in order to generate a triangular mesh at each time step. We have used the algorithm described in [5]. The initial mesh has: 208 vertices, 362 triangles, $M = 32$ (the number of vertices on the boundary Γ_1), $h = 0.155178$ (the mesh size).

The number of vertices, triangles, boundary edges are the same for the first 10 meshes. In the below table, we see the evolution of the mesh size h.

Time step (n)	1	2	3	4	5
Mesh size (h)	0.155	0.171	0.187	0.202	0.216

Time step (n)	6	7	8	9	10
Mesh size (h)	0.231	0.244	0.256	0.266	0.276

After 10 time steps, the domain might be a semicircle of ray $e^{0.5} \approx 1.648721$.

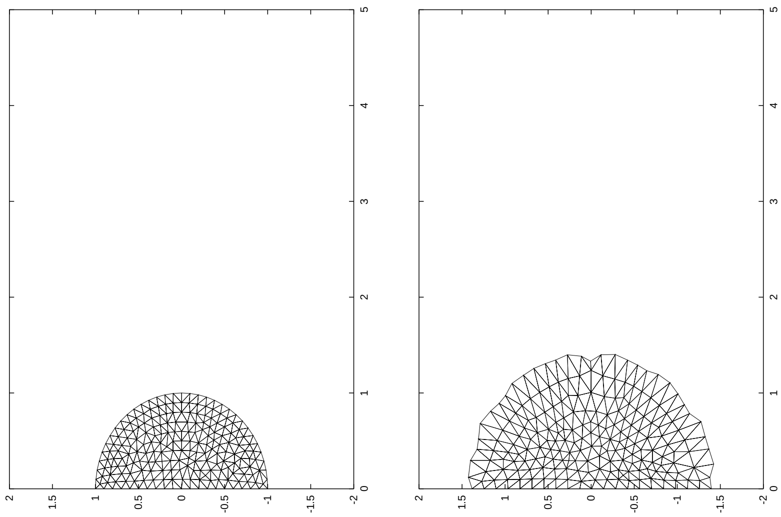


Figure 1: The initial mesh and the mesh after 10 time steps

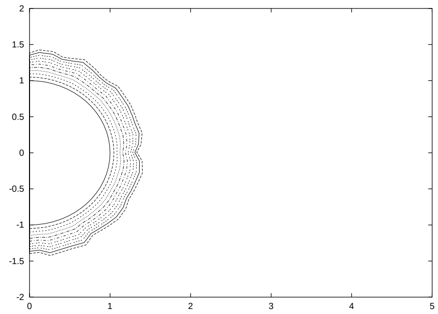


Figure 2: The evolution of the moving boundary (0–10 time steps)

The second simulation was performed using the **Algorithm 2** for $\Delta t = 0.0005$ and $N = 1000$ (the number of time steps). The others parameters are the same as in the first simulation.

The software `freefem+` [1] was used to generate a triangular mesh at each time step. The numbers of the vertices and of the triangles are not the same for the all meshes. For example, the mesh after 1000 time steps has 205 vertices and 356 triangles.

After 1000 time steps, the domain might be a semicircle of ray $e^{0.5} \approx 1.648721$.

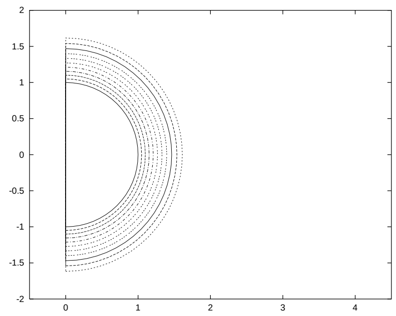


Figure 3: The boundary after 0, 100, . . . , 1000 time steps

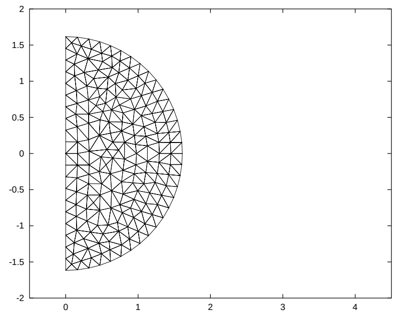


Figure 4: The mesh after 1000 time steps

References

[1] D. Bernardi, F. Hecht, K. Ohtsuka, O. Pironneau, A finite element software for PDE: freefem+, `http://www-rocq.inria.fr/Frederic.Hecht`

[2] J. Escher, G. Simonett, Classical solution of multidimensional Hele-Shaw models, *SIAM J. Math. Anal.*, **28** (1997), No. 5, 1028–1047

[3] F. Hecht, C++ et Eléments Finis, Cours DEA, Université Paris VI, `http://www-rocq.inria.fr/Frederic.Hecht`

[4] T.Y. Hou, J.S. Lowengrub, M.J. Shelley, Removing the stiffness from interfacial flows with surface tension, *J. Comput. Phys.*, **114** (1994), 312–328

[5] C.M. Murea, Dynamic meshes generation using the relaxation method with applications to fluid-structure interaction problems, *An. Univ. Bucuresti Mat.*, **47** (1998), No. 2, 177–186

[6] O. Pironneau, F. Hecht, Introduction au Calcul Scientifique en C++, Cours Matrîse de Mathématiques, Ingénierie Mathématique, Université Paris VI, `http://www-rocq.inria.fr/Frederic.Hecht`