5.7 Level Sets and the Fast Marching Method

The level sets of $f(x,y)$ are the sets on which the function is constant. For example $f(x,y) = x^2+y^2$ is constant on circles around the origin. Geometrically, a level plane $z =$ constant will cut through the surface $z = f(x,y)$ on a level set. One attractive feature of working with level sets is that their topology can change (pieces of the level set can separate or come together) just by changing the constant.

Starting from one level set, the **signed distance function** $d(x,y)$ is especially important. It gives the *distance* to the level set, and also the sign: typically $d > 0$ outside and $d < 0$ inside. For the unit circle, $d = r - 1 = \sqrt{x^2+y^2} - 1$ will be the signed distance function. In the mesh generation algorithm of Section 2.____, it was convenient to describe the region by its distance function $d(x,y)$.

A fundamental fact of calculus: The gradient of $f(x,y)$ is perpendicular to its level sets. Reason: In the tangent direction t to the level set, $f(x,y)$ is not changing and $(\operatorname{grad} f)\cdot t$ is zero. So $\operatorname{grad} f$ is in the normal direction. For the function x^2+y^2, the gradient $(2x,2y)$ points outward from the circular level sets. The gradient of $d(x,y) = \sqrt{x^2+y^2} - 1$ points the same way, and it has a special property: **The gradient of a distance function is a unit vector**. It is the *unit* normal $n(x,y)$ to the level sets. For the circles,

$$\operatorname{grad}(\sqrt{x^2+y^2}-1) = \left(\frac{x}{r},\frac{y}{r}\right) \qquad \text{and} \qquad |\operatorname{grad}|^2 = \frac{x^2}{r^2}+\frac{y^2}{r^2} = 1\,. \tag{1}$$

You could think of the level set $d(x,y) = 0$ as a wall of fire. This firefront will move normal to itself. If it has constant velocity 1 then at time T the fire will reach all points on the level set $d(x,y) = T$.

That "wall of fire" example brings out an important point when the zero level set has a corner (it might be shaped like a **V**). The points at distance d outside that set (the firefront at time d) will lie on lines parallel to the sides of the **V**, and also on a circular arc of radius d around the corner. For $d < 0$ the **V** moves *inward*. It remains a **V** (with no smoothing of the corner).

The central problem of the level set method is to **propagate a curve** like the firefront. A velocity field $v = (v_1, v_2)$ gives the direction and speed of each point for the movement. At time $t = 0$, the curve is the level set where $d(x,y) = 0$. At later times the curve is the zero level set of a function $\phi(x,y,t)$. The fundamental **level set equation** in its first form is

$$\frac{d\phi}{dt} + v\cdot\operatorname{grad}\phi = 0, \quad \text{with } \phi = d(x,y) \text{ at } t = 0\,. \tag{2}$$

In our wall of fire example, v would be the unit vector in the normal direction to the firefront: $v = n = \operatorname{grad}\phi/|\operatorname{grad}\phi|$. In all cases it is only the normal component $F = v\cdot n$ that moves the curve! Tangential movement (like rotating a circle around its center) gives no change in the curve as a whole. By rewriting $v\cdot\operatorname{grad}\phi$, the level set equation takes a second form that is more useful in computation:

$$v \cdot \operatorname{grad}\phi = v \cdot \frac{\operatorname{grad}\phi}{|\operatorname{grad}\phi|}|\operatorname{grad}\phi| = F|\operatorname{grad}\phi| \quad \text{leads to} \quad \frac{d\phi}{dt} + F|\operatorname{grad}\phi| = 0\,. \tag{3}$$

We only need to know the velocity field v (and only its normal component F) near the *current location* of the level curve–not everywhere else. We are propagating a curve. The velocity field may be fixed (easiest case) or it may depend on the local shape of the curve (nonlinear case). An important example is **motion by mean curvature**: $F = -\kappa$. The neat property $|\operatorname{grad}\phi| = 1$ of distance functions simplifies the formulas for the normal n and curvature κ:

When ϕ is a distance function

$$n = \frac{\operatorname{grad}\phi}{|\operatorname{grad}\phi|} \quad \text{becomes} \quad n = \operatorname{grad}\phi$$
$$\kappa = \operatorname{div} n \quad \text{becomes} \quad \kappa = \operatorname{div}(\operatorname{grad}\phi) = \text{Laplacian of } \phi \tag{4}$$

But here is an unfortunate point for $t > 0$. Constant speed ($F = 1$) in the normal direction does maintain the property $|\operatorname{grad}\phi| = 1$ of a distance function. Motion by mean curvature, and other motions, will destroy this property. To recover the simple formulas (4) for distance functions, the level set method often **reinitializes** the problem—restarting from the current time t_0 and computing the distance function $d(x, y)$ to the current level set $\phi(x, y, t_0) = 0$. This reinitialization was the **Fast Marching Method**, which finds distances from nearby meshpoints to the current level set.

We describe this quick method to compute distances to meshpoints, and then discuss the numerical solution of the level set equation (3) on the mesh.

Fast Marching Method

The problem is to march outward, computing distances from meshpoints to the interface (the current level set where $\phi = 0$). Imagine that we know these distances for the grid points adjacent to the interface. (We describe fast marching but not the full algorithm of reinitialization.) The key step is to compute the distance to the *next nearest meshpoint*. Then the front moves further outward with velocity $F = 1$. When the front crosses a new meshpoint, it will become the next nearest and its distance will be settled next.

So we accept one meshpoint at a time. Distances to further meshpoints are tentative (not accepted). They have to be recomputed using the newly accepted meshpoint and its distance. The Fast Marching Method must quickly take these steps recursively:

1. Find the tentative meshpoint p with smallest distance (to be accepted).

2. Update the tentative distances to all meshpoints adjacent to p.

To speed up step **1**, we maintain a binary tree of unaccepted meshpoints and their tentative distances. The smallest distance is at the top of the tree, which identifies p. When that value is removed from the tree, others move up to form the new tree.

Recursively, each vacancy is filled by the smaller of the two distance values below it. Then step **2** updates those values at points adjacent to p. These updated values may have to move (a little) up or down to reset the tree. In general, the updated values should be smaller (they mostly move up, since they have the latest meshpoint p as a new candidate in finding the shortest route to the original interface).

The Fast Marching Method finds distances to N meshpoints in time $O(N \log N)$. The method applies when the front moves *in one direction only*. The underlying equation is $F|\nabla T| = 1$ (Eikonal equation with $F > 0$). The front never crosses a point twice (and the crossing time is T). If the front is allowed to move in both directions, and F can change sign, we need the initial value formulation (3).

Lagrangian versus Eulerian

A fundamental choice in analyzing and computing fluid flow is between Lagrange and Euler. For the minimizing function in optimization, they arrived at the same "Euler-Lagrange equation". In studying fluids, they chose *very different* approaches:

Lagrange follows the path of each particle of fluid. He moves.

Euler sees which particles pass through each point. He sits.

Lagrange is more direct. He "*tracks*" the front. At time zero, points on the front have positions $x(0)$. They move according to vector differential equations $dx/dt = V(x)$. If we mark and follow a finite set of points, equally spaced at the start, serious difficulties can appear. Their spacing can get very tight or very wide (forcing us to remove or add marker points). The initial curve can split apart or cross itself (changes of topology). The level set method escapes from these difficulties by going Eulerian.

For Euler, the x-y coordinate system is fixed. He "*captures*" the front implicitly, as a level set of $\phi(x, y, t)$. When the computational grid is also fixed, we are constantly interpolating to locate level sets and compute distance functions. Squeezing or stretching or tangling of the front appear as changes in ϕ, not as disasters for the mesh.

The velocity v *on the interface* determines its movement. When the level set method needs v at a meshpoint off the interface, a good candidate is the value of v at the nearest point on the interface.

Upwind Differencing

The level set finite difference method is properly developed in the books by its originators: Sethian [ ] and Osher and Fedkiw [ ]. Here we concentrate on an essential point:

upwind differencing. *Recall* from Section 1.____ the three simplest approximations **F**, **B**, **C**to the first derivative $d\phi/dx$–*Forward*, *Backward*, *Centered*:

$$\mathbf{F} \quad \frac{\phi(x+h)-\phi(x)}{h} \qquad \mathbf{B} \quad \frac{\phi(x)-\phi(x-h)}{h} \qquad \mathbf{C} \quad \frac{\phi(x+h)-\phi(x-h)}{2h}$$

Which do we use in the simple convection equation $d\phi/dt + a\, d\phi/dx = 0$? Its true solution is $\phi(x-at,0)$. *The choice of finite differences depends on the sign of a.* The flow moves left to right for $a < 0$. Then the *backward* difference is natural—the "upwind" value $\phi(x-h,t)$ on the left should contribute to $\phi(x,t+\Delta t)$. The downwind value $\phi(x+h,t)$ on the right moves further downwind during the time step, and has no influence at x.

When the movement of the solution (and the wind) is right to left (with $a > 0$), then the *forward* difference will use the appropriate upwind value $\phi(x+h,t)$ along with $\phi(x,t)$, in computing the new $\phi(x,t+\Delta t)$.

Notice the time-step limitation $|a|\Delta t \leq h$. In time Δt, the "wind" will bring the true value of ϕ from $x + a\Delta t$ to the point x. If $a > 0$ and finite differences reach upwind to $x+h$, that must be far enough to include information at $x + a\Delta t$. So the *Courant-Friedrichs-Lewy condition* is $a\Delta t \leq h$. The numerical waves must propagate at least as fast as the physical waves (and in the right direction!). Downwind differencing is looking for that information on the wrong side of the point x, and is doomed to failure. Centered differencing in space is unstable for ordinary forward Euler.

By careful choice of the right finite differences, Osher has constructed higher-order essentially non-oscillatory (ENO) schemes. A central idea in nonlinear problems, where the differential equation has multiple solutions (see Section ____), is to choose the "*viscosity solution*." This physically correct solution appears in the limit as an extra ϵu_{xx} diffusion term goes to zero. With good differencing the viscosity solution is the one that appears as $\Delta x \to 0$.

At this point, the level set method does not appear in large production codes. In research papers it has successfully solved a great variety of difficult nonlinear problems.