1 Numerical Simulation

Our numerical simulation of the quasi-static interface evolution is based on a combination of the level set method and the finite element method. The deformed interface is represented implicitly and propagated using the level set method. The elastostatic problem is discretized with a finite element method, using the same grid as the interface representation. A highly efficient matrix-free multigrid solver is used to solve the linear systems of equations, which allows us to use millions of degrees of freedom on a standard desktop computer.

1.1 Interface Evolution

The level set method of Osher and Sethian [5], [7], [4] is a technique for modeling moving interfaces. It has found widespread use in the simulation of various physical problems and it handles interfaces with very large deformations, even with topological changes. In our application, the main advantages compared to explicit techniques are the stability under curvature motion and the straightforward extension to three dimensions.

The interface is represented by the zero level set $\phi(\boldsymbol{x}) = 0$ of a function ϕ discretized on a Cartesian grid, see Fig. 1 for a planar example. We also require that ϕ is a signed distance function, with negative sign inside our domain, positive outside, and $|\nabla\phi| = 1$. Note that there is no explicit representation of the interface, but we can compute geometric quantities directly from ϕ, such as the normal vector $\boldsymbol{n} = \nabla\phi/|\nabla\phi|$ and the mean curvature $\kappa = \nabla\cdot(\nabla\phi/|\nabla\phi|)$ (we do not substitute $|\nabla\phi| = 1$ into these expressions, because ϕ will generally not be an exact distance function). These expressions are defined for all $\boldsymbol{x}$, not just those on the interface, which gives us a smooth embedding of the quantities in the entire computational domain.

We can now evolve the interface according to a given normal velocity field F, by numerically solving a Hamilton-Jacobi equation. The *level set equation*

$$\phi_t + F|\nabla\phi| = 0 \tag{1}$$

models this evolution, where F depends on the geometry of the interface, the solutions of the underlying physical problem, as well as any other dependence of space and time.

During the evolution, ϕ will in general not remain a signed distance function. Initially, this is not a problem for the accuracy of the simulations, since none of the expressions assume $|\nabla\phi| = 1$. But after some time, the level sets of ϕ will be highly deformed, and then the numerical accuracy drops. To avoid this, we *reinitialize* ϕ regularly, by solving for a new ϕ satisfying $|\nabla\phi| = 1$ but with the same zero level set as before. In [11], this was done by solving a nonlinear PDE (the *reinitialization equation*), but this technique perturbs the interface significantly.

Instead, we calculate the distances explicitly for all cells close to the interface. For each cell, we detect the interface and compute the signed distances to the neighboring grid nodes. The remaining nodes are updated by the *fast marching*

method [6], which solves the Eikonal equation $|\nabla\phi| = 1$ in almost linear time, making ϕ a signed distance function in the entire domain.

In our application, the normal velocity field F is generally not mass conserving. We implement the conservation constraint by solving for a Lagrange multiplier ν such that the velocity field $F + \nu$ conserves mass. This is a nonlinear problem with only one unknown ν, and we solve it using Newton's method. Note that we can compute the volume of the geometry directly from ϕ according to $V = \int(1 - \theta(\phi(x)))\,dx$, where $\theta(x)$ is Heaviside's step function. We never use any explicit representation of the interface, except for the reinitialization and the postprocessing.

For the numerical solution of (1), we use different schemes for the motion due to curvature and for the general, curvature independent motion. For the general motion, we use a first order finite difference approximation on the Cartesian grid:

$$\phi_{ijk}^{n+1} = \phi_{ijk}^{n} + \Delta t_1 \left(\max(F,0)\nabla_{ijk}^{+} + \min(F,0)\nabla_{ijk}^{-} \right), \tag{2}$$

where

$$\begin{aligned} \nabla_{ijk}^{+} = \big[& \max(D^{-x}\phi_{ijk}^{n},0)^2 + \min(D^{+x}\phi_{ijk}^{n},0)^2 + \\ & \max(D^{-y}\phi_{ijk}^{n},0)^2 + \min(D^{+y}\phi_{ijk}^{n},0)^2 + \\ & \max(D^{-z}\phi_{ijk}^{n},0)^2 + \min(D^{+z}\phi_{ijk}^{n},0)^2 \big]^{1/2}, \end{aligned} \tag{3}$$

$$\begin{aligned} \nabla_{ijk}^{-} = \big[& \min(D^{-x}\phi_{ijk}^{n},0)^2 + \max(D^{+x}\phi_{ijk}^{n},0)^2 + \\ & \min(D^{-y}\phi_{ijk}^{n},0)^2 + \max(D^{+y}\phi_{ijk}^{n},0)^2 + \\ & \min(D^{-z}\phi_{ijk}^{n},0)^2 + \max(D^{+z}\phi_{ijk}^{n},0)^2 \big]^{1/2}. \end{aligned} \tag{4}$$

Here, D^{-x} is the backward difference operator in the x-direction, D^{+x} the forward difference operator, etc, see [7] for more details. We use a first order scheme of a similar form in the fast marching method used for the reinitialization.

For the curvature dependent part of F, we use central differencing for all difference approximation and update ϕ according to:

$$\phi_{ijk}^{n+1} = \phi_{ijk}^{n} + \Delta t_2 \frac{\left\{ \begin{array}{c} (\phi_{yy} + \phi_{zz})\phi_x^2 + (\phi_{xx} + \phi_{zz})\phi_y^2 + (\phi_x x + \phi_y y)\phi_z^2 \\ -2\phi_x\phi_y\phi_{xy} - 2\phi_x\phi_z\phi_{xz} - 2\phi_y\phi_z\phi_{yz} \end{array} \right\}}{\phi_x^2 + \phi_y^2 + \phi_z^2}. \tag{5}$$

The timestep has to be small for this scheme to be stable, and we take several steps with (5) for each step with (3), with a corresponding reduction in the timestep Δt_2.

1.2 Linear Elasticity

We discretize the elastostatic equations using the finite element method (FEM) [9]. Our mesh is the same Cartesian grid as we use for representing ϕ, and

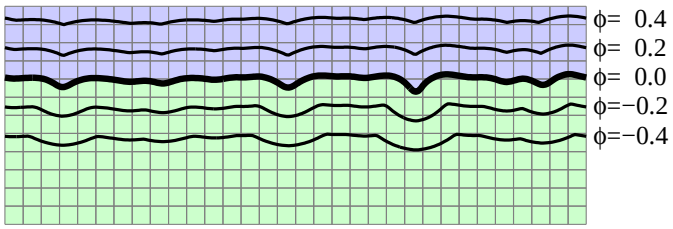


Figure 1: The level set representation of a deformed two-dimensional geometry. The signed distance function ϕ is represented on a Cartesian grid, and the interface is given by $\phi(x, y) = 0$.

all elements are cubes of the same size. The finite element method is often associated with unstructured meshes that conform to the boundary, but here we use it in a way resembling finite difference discretizations.

One important issue is how to treat the deformed surface. The physical condition is zero normal stress, which corresponds to a Neumann boundary condition. Since our grid is not aligned with the surface, we can not rely on traditional FEM for these conditions. One solution is to use the immersed interface method, which enforces the boundary conditions as constraints on the grid nodes close to the interface [3], [8]. Another method is to generate a new, unstructured mesh and solve the problem on this mesh, and then interpolate back to the Cartesian grid for the interface propagation [10] . We use a simple approach, sometimes referred to as the Ersatz method [1], where we solve the elastostatic equation in the entire domain, and set Young's modulus E to a small number outside the actual geometry. This is a good approximation to Neumann conditions, the accuracy is high enough for our purposes, and since we use a multigrid method we do not suffer from disadvantages such as ill-conditioning of the stiffness matrix.

We use first order shape functions and Gauss integration to compute the local stiffness matrices. Note that all of the elements are geometrically identical, and most of them have constant material properties, which we take advantage of to speed up the assembly process. All the elements around the interface have to be assembled separately, and we use a smoothed step function for Young's modulus and integrate with a high-order scheme.

To solve the discretized linear system of equations we use a multigrid method with under-relaxed Jacobi iterations for the smoothing. With a Cartesian grid, the grid hierarchy is trivial to create (as long as the number of cells in each direction are divisible by powers of two). Using an iterative solver also means we always have good initial guesses from the last time step. The Dirichlet conditions and the periodicity are implemented as constraints on the system, and we eliminate the corresponding degrees of freedom.

The prestraining $\varepsilon_x, \varepsilon_y$ is applied on the discretized system, by writing the total displacement field as a sum of a given stretched field $U_0 = \varepsilon_x X + \varepsilon_y Y$ and

an unknown, periodic perturbation U. We then solve for U in $KU = -KU_0$, where K is the stiffness matrix with boundary conditions incorporated.

To obtain the required resolution, we need a very large number of nodes. For example, a grid of size $129 \times 129 \times 65$ gives more than a million nodes, more than 3 million unknowns, and just to store this solution in computer memory requires 26MB. Furthermore, the stiffness matrix will have an average of 81 elements per row, and storing it in compressed column format [2] would require 3GB of memory. Clearly, to solve our problems accurately on a desktop computer we have to use a different technique.

Since most of the elements are either inside or outside the domain and have identical geometry, we can compute two local stiffness matrices which can be reused many times. Therefore, we do not create and store the global stiffness matrix, but we repeat the assembly process every time we operate with the matrix. This will result in more operations than a simple sparse matrix multiplication, but it allows us to solve very large problems. The two operations we perform on the matrix are multiplication by a vector for the residual, and the under-relaxed Jacobi step.

1.3 Results

Figure 2 shows the results of a three dimensional simulation. Our computational domain is a block of dimensions $4\times4\times1$, discretized with a grid of size $193\times193\times49$ for a total of $1,825,201$ nodes and $5,475,603$ degrees of freedom. Initially, the surface is located a distance 0.66 from the bottom of the domain, and the height is perturbed at each node in the x, y-plane by normal distributed random numbers with standard deviation 0.0025. The material has Young's modulus $E = 1$, Poisson's ratio $\nu = 0.3$, and surface tension $\sigma = 0.10$ and $\sigma = 0.05$ in the two simulations.

The boundary conditions specify the displacement in the z-direction $w = 0$ at the bottom face, and all displacements u, v, w are periodic at the left/right and the front/back faces. We use a timestep $\Delta t_1 = 0.05\sigma$ for the curvature independent part, and $\Delta t_2 = \Delta t_1/10$ for the motion by curvature.

The two plots in Figure 2 show the surface at times $t = 240\Delta t_1$ and $t = 350\Delta t_1$, respectively. The color represents the elastic energy density, ranging from zero (blue) to large values (red). Animations of the quasi-static time evolution can be found at www-math.mit.edu/∼persson/qdots.

References

[1] G. Allaire, F. Jouve, and A.-M. Toader. A level-set method for shape optimization. *C.R. Acad. Sci. Paris, Ser. I*, 334:1125–1130, 2002.

[2] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Transactions on Mathematical Software*, 15(1):1–14, March 1989.

[3] R. J. LeVeque and Z. Li. The immersed interface method for elliptic equations with discountinuous coefficients and singular sources. *SIAM J. Numer. Anal.*, 31:1019, 1994.

[4] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[5] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. of Computational Physics*, 79:12–49, 1988.

[6] J. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proceedings of the National Academy of Sciences*, volume 93 (4), pages 1591–1595, 1996.

[7] J. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry*. Cambridge University Press, 1999.

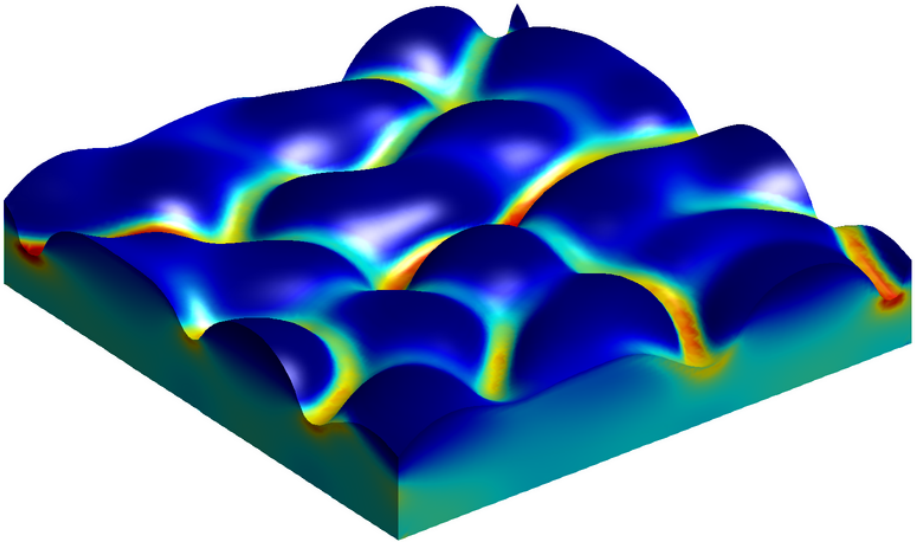
[8] J. Sethian and A. Wiegmann. Structural boundary design via level set and immersed interface methods. *J. Comput. Phys.*, 163:489–528, 2000.

[9] G. Strang and G. Fix. *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press, 1973.

[10] G. Strang and P.-O. Persson. Circuit simulation and moving mesh generation. In *Proc. of Int. Symp. on Comm. and Inform. Tech. 2004 (ISCIT 2004)*, November 2005.

[11] M. Sussman, P. Smereka, and S. Osher. A levelset approach for computing solutions to incompressible two-phase flow. *J. of Computational Physics*, 114:146–159, 1994.

Final Configuration, $\sigma = 0.10$



Final Configuration, $\sigma = 0.05$

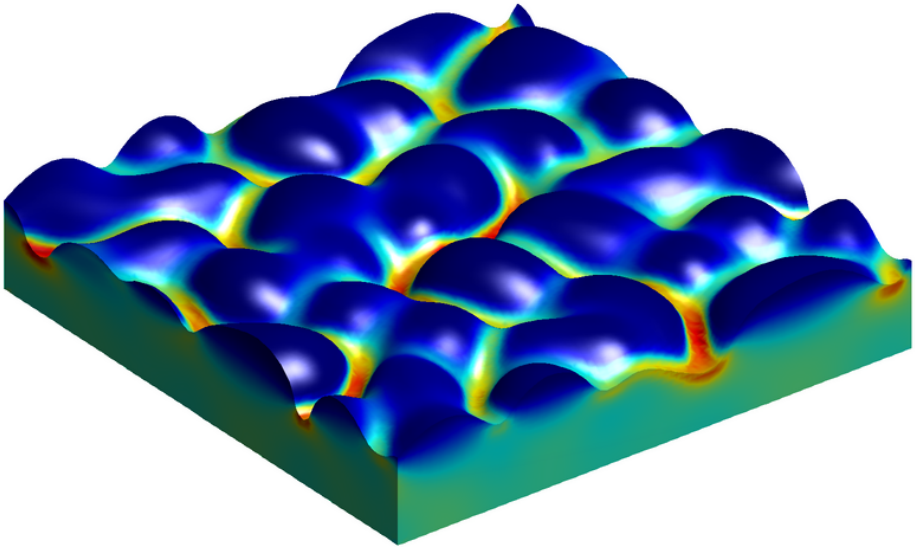


Figure 2: Results of the 3-D simulation.