

Numerical Simulation in 2-D martensitic phase transition and microstructure

Weigang Zhong
Advisor: Bo Li

Abstract

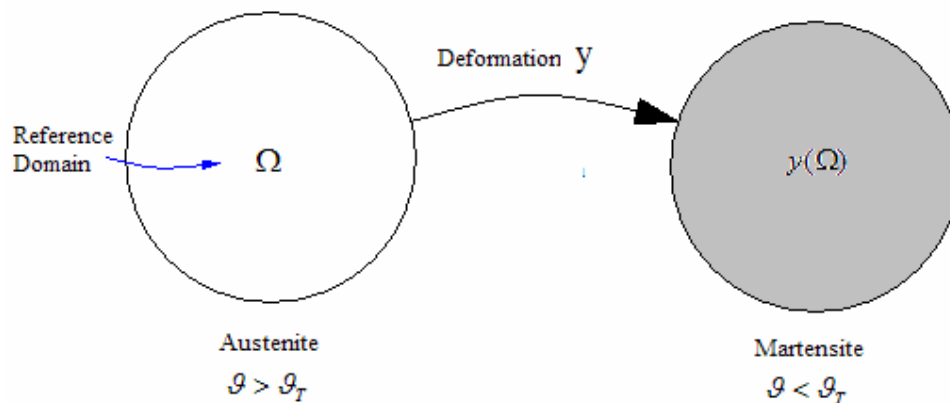
This project studies numerically the two-dimensional (2D) martensitic phase transition and microstructure. A martensitic microstructure is a fine-scale mixture of different martensitic variants that can often be observed during a martensitic phase transition. A mathematical model for martensitic microstructure is the minimization of a multi-well energy functional of deformations that satisfy certain boundary conditions.

1. Description of Martensitic Phase Transition and Microstructure

I am interested in the transformation of high temperature austenitic microstructure to low temperature martensitic microstructure. Briefly, an austenitic microstructure is the homogeneous microstructure, a constituent of alloys that is formed when atoms of an element are incorporated into the crystals of a metal. Martensitic microstructure is formed by the breakdown of austenite. Usually we have the martensitic twin in the material, which involves two variants together.

Austenite: A high-temperature phase with stable lattice structure, this phase only has one variant.

Martensite: A low-temperature phase with stable nonsymmetrical structure, this phase appears in several variants.



2. A Mathematical Model

The underlying problem is to minimize the bulk energy. Mathematically I need to set up the nonconvex free energy functions with some given density functions, and discretize the domain, use finite element method to minimize the energy function.

I concentrate on problem of two dimension mathematical model.

Say, we have a reference domain: $\Omega \subset R^n$ (undeformed austenite)

At a fixed temperature $\mathcal{G} < \mathcal{G}_t$, let's define Energy Density $\phi(F)$, where $F = \nabla y \in R^{n \times n}$, deformation

$$y : \Omega \rightarrow R^n$$

- $\phi(F) \geq 0$ with $\phi(U_0) = \phi(U_1) = 0$.

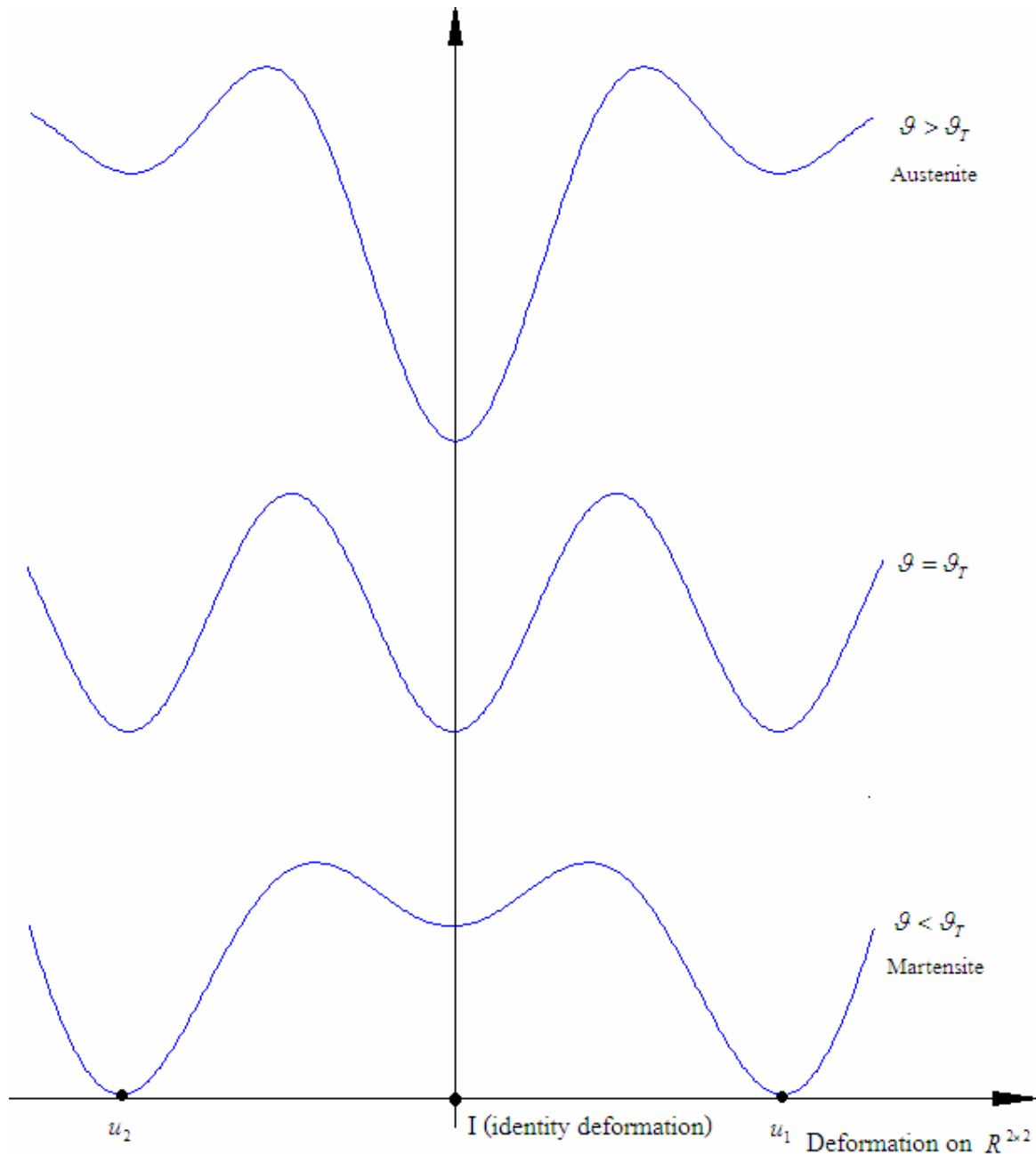
$$\phi(F) = 0 \Leftrightarrow F \in SO(3)U_1 \cup \dots \cup SO(3)U_N \text{ (energy minimizer)}$$

$SO(3)$ is the set of all proper rotations; $SO(3)U_i (i = 1, \dots, N)$: energy well

- $\phi(RF) = \phi(F)$, for all rotation R . (Frame indifference)
- $\phi(FR_i) = \phi(F)$, $\forall R \in g = \{R_1, \dots, R_s\}$ is a symmetry group. (Material symmetry)
- Minimizing gradients $RU_i =$ variants of martensite.
- $\phi(F) \geq \alpha \|F - \Pi F\|^p$, $p > 1$, Π projects onto variants.
- $\phi(F) \geq 0$ with $\phi(U_0) = \phi(U_1) = 0$.

In case that ϕ has the so-called two-well structure, it means that ϕ attains its minimum precisely in two equilibrium states represented by deformation tensors F_1 and F_2 such that $F_1 \neq RF_2$ for any rotation $R \in SO(2)$.

Here are graphic demos of the Two-Well density:



The problem I am working on is a Two-Well problem with a Two-Well density.

This is a two-dimensional energy model. The total energy is associated with a deformation $y(x)$, of the reference domain $\Omega = [0,1]^2$, $y : \Omega \rightarrow \mathbb{R}^{2 \times 2}$. I consider Lipschitz continuous functions which satisfy the boundary condition $y(x) = F(x)$. The energy function is given by

$$E(y) = \int_{\Omega} W(\nabla y(x)) dx$$

where $W(F)$ is the energy density. Take $W(F) = \Phi(F^T F) = \Phi(C)$ and

$$\Phi(C) = \kappa_1 (\text{tr}C - 2)^2 + \kappa_2 C_{12}^2 + \kappa_3 \left(\left(\frac{C_{11} - C_{12}}{2} \right)^2 - \varepsilon^2 \right)^2$$

with constants $\kappa_i > 0$ and $\varepsilon > 0$,

The form of the energy is based on a suggestion by Ericksen and James and is based on Ericksen's work on constrained elastic crystals.

The numerical method of this problem is to take a finite element approximation of the space, and to minimize the energy over the finite dimensional space using an iterative descent method.

3. Finite Element Discretization

For the reference domain $\Omega = [0,1]^2$, we use the uniform square mesh with $h=1/N$ being the size of the mesh.

The finite elements are

$$\Omega_{ij} = [ih, (i+1)h] \times [jh, (j+1)h]$$

and the nodes (ih, jh) .

We may define the bilinear finite elements

$$\mathcal{Q}_h = \{v \in C^0(\Omega; \mathbb{R}^2) : v|_{\Omega_{ij}} \text{ is bilinear in each element for each } i \text{ and } j\}$$

Since this is a two-dimension problem, for $u_h \in \mathcal{Q}_h$, we have $u_h = (u_h^1, u_h^2)$ with

$$u_h^k = \sum_{i,j=0}^{N-1} u_{ij}^k \phi_{ij}, \quad k = 1 \text{ or } 2$$

where ϕ_{ij} are the bilinear basis functions.

By the discretization above, I can write down the approximation of the energy function

$$E(y) = h^2 \sum_{i,j=0}^{N-1} W(\nabla u_h(m_{ij}))$$

where m_{ij} is the midpoint of Ω_{ij} .

One problem is how to find ∇u_h ? The answer is to use the basis bilinear functions.

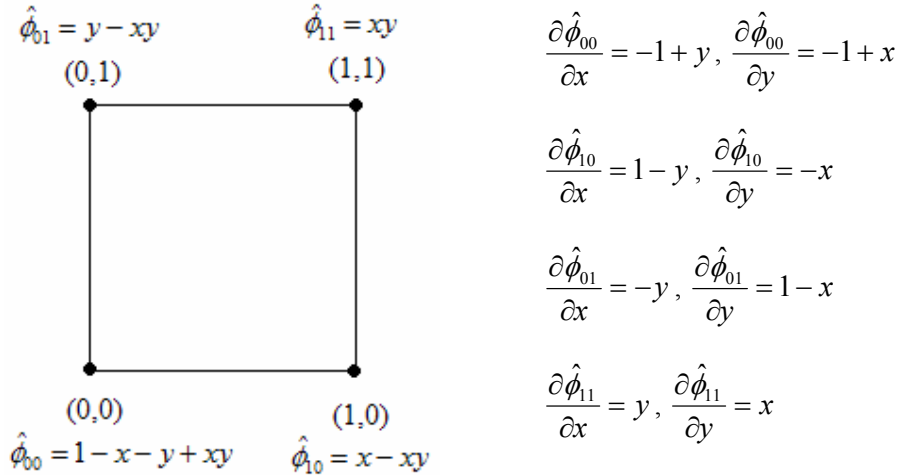
$$\text{In } \Omega_{ij}, u_h^k = u_{ij}^k \phi_{ij} + u_{i+1,j}^k \phi_{i+1,j} + u_{i,j+1}^k \phi_{i,j+1} + u_{i+1,j+1}^k \phi_{i+1,j+1}; \nabla u_h = \begin{pmatrix} \frac{\partial u_h^1}{\partial x} & \frac{\partial u_h^1}{\partial y} \\ \frac{\partial u_h^2}{\partial x} & \frac{\partial u_h^2}{\partial y} \end{pmatrix}$$

where

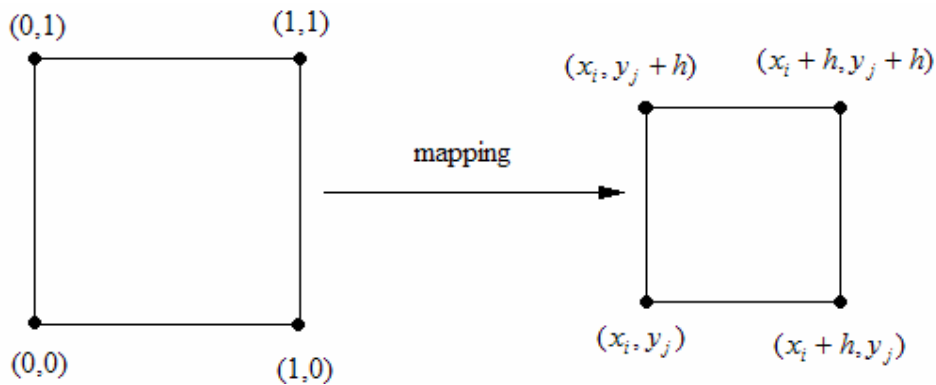
$$\frac{\partial u_h^k}{\partial x} = u_{ij}^k \frac{\partial \phi_{ij}}{\partial x} + u_{i+1,j}^k \frac{\partial \phi_{i+1,j}}{\partial x} + u_{i,j+1}^k \frac{\partial \phi_{i,j+1}}{\partial x} + u_{i+1,j+1}^k \frac{\partial \phi_{i+1,j+1}}{\partial x}$$

$$\frac{\partial u_h^k}{\partial y} = u_{ij}^k \frac{\partial \phi_{ij}}{\partial y} + u_{i+1,j}^k \frac{\partial \phi_{i+1,j}}{\partial y} + u_{i,j+1}^k \frac{\partial \phi_{i,j+1}}{\partial y} + u_{i+1,j+1}^k \frac{\partial \phi_{i+1,j+1}}{\partial y} \quad k = 1 \text{ or } 2$$

Consider a unitary element



There is a mapping from the unitary square to arbitrary square area



$$\frac{\partial \phi_{ij}}{\partial x} = -\frac{1}{h} + \frac{y-y_j}{h^2}; \frac{\partial \phi_{i+1,j}}{\partial x} = \frac{1}{h} - \frac{y-y_j}{h^2}; \frac{\partial \phi_{i,j+1}}{\partial x} = -\frac{y-y_j}{h^2}; \frac{\partial \phi_{i+1,j+1}}{\partial x} = \frac{y-y_j}{h^2}$$

$$\frac{\partial \phi_{ij}}{\partial y} = -\frac{1}{h} + \frac{x - x_i}{h^2}, \quad \frac{\partial \phi_{i+1j}}{\partial y} = -\frac{x - x_i}{h^2}, \quad \frac{\partial \phi_{ij+1}}{\partial y} = \frac{1}{h} - \frac{x - x_i}{h^2}, \quad \frac{\partial \phi_{i+1j+1}}{\partial y} = \frac{x - x_i}{h^2}$$

Therefore, we can find $\nabla u_h(m_{ij})$, where $m_{ij} = (x_i + h/2, y_j + h/2)$

$$\frac{\partial u_h^k}{\partial x}(m_{ij}) = \frac{1}{2h} (u_{i+1j}^k + u_{i+1j+1}^k - u_{ij}^k - u_{ij+1}^k)$$

$$\frac{\partial u_h^k}{\partial y}(m_{ij}) = \frac{1}{2h} (u_{ij+1}^k + u_{i+1j+1}^k - u_{ij}^k - u_{i+1j}^k)$$

Now we can evaluate the energy function for the given domain and density.

If we want to use the numerical method with the computation of first derivatives to minimize the energy function we just got, it is necessary for me to find ∇E . We can compute the components of ∇E using the bilinear basis functions ϕ_{ij} in each element Ω_{ij} .

Here I explicitly give the partial derivatives of density with respect to each node of one element:

$$\frac{\partial W_{ij}}{\partial u_{ij}^k} = -\frac{1}{h} \left[[(I) + (II)] \left(\frac{\partial u_h^k}{\partial x} + \frac{\partial u_h^k}{\partial y} \right) + (III) \left(\frac{\partial u_h^k}{\partial x} - \frac{\partial u_h^k}{\partial y} \right) \right]$$

$$\frac{\partial W_{ij}}{\partial u_{ij+1}^k} = -\frac{1}{h} \left[[(I) - (II)] \left(\frac{\partial u_h^k}{\partial x} - \frac{\partial u_h^k}{\partial y} \right) + (III) \left(\frac{\partial u_h^k}{\partial x} + \frac{\partial u_h^k}{\partial y} \right) \right]$$

$$\frac{\partial W_{ij}}{\partial u_{i+1j}^k} = \frac{1}{h} \left[[(I) - (II)] \left(\frac{\partial u_h^k}{\partial x} - \frac{\partial u_h^k}{\partial y} \right) + (III) \left(\frac{\partial u_h^k}{\partial x} + \frac{\partial u_h^k}{\partial y} \right) \right]$$

$$\frac{\partial W_{ij}}{\partial u_{i+1j+1}^k} = \frac{1}{h} \left[[(I) + (II)] \left(\frac{\partial u_h^k}{\partial x} + \frac{\partial u_h^k}{\partial y} \right) + (III) \left(\frac{\partial u_h^k}{\partial x} - \frac{\partial u_h^k}{\partial y} \right) \right]$$

where

$$(I) = 2\kappa_1 \left(\left(\frac{\partial u_h^1}{\partial x} \right)^2 + \left(\frac{\partial u_h^2}{\partial x} \right)^2 + \left(\frac{\partial u_h^1}{\partial y} \right)^2 + \left(\frac{\partial u_h^2}{\partial y} \right)^2 - 2 \right)$$

$$(II) = \kappa_2 \left(\frac{\partial u_h^1}{\partial x} \frac{\partial u_h^1}{\partial y} + \frac{\partial u_h^2}{\partial x} \frac{\partial u_h^2}{\partial y} \right)$$

$$(III) = 2\kappa_3 \left(\left(\frac{\left(\frac{\partial u_h^1}{\partial x} \right)^2 + \left(\frac{\partial u_h^2}{\partial x} \right)^2 - \left(\frac{\partial u_h^1}{\partial y} \right)^2 - \left(\frac{\partial u_h^2}{\partial y} \right)^2}{2} \right)^2 - \varepsilon^2 \right)$$

$\frac{\partial u_h^k}{\partial x}$ & $\frac{\partial u_h^k}{\partial y}$ are given above.

Since $E(y) = h^2 \sum_{i,j=0}^{N-1} W(\nabla u_h(m_{ij}))$, we can simply sum up all $\frac{\partial W_{ij}}{\partial u_{ij}^k}$, multiplied by h^2 , then we can get each component of ∇E .

Things above are the discretization format of my Two-Well problem by using the Finite Element. The rest of things are to design the numerical method to minimized the energy function.

4. Numerical Optimization

Given a single function that depends on one or more independent variables, we want to find the value of those variables where the function takes on a maximum or a minimum value. Our computational effort is to find a quick cheap numerical method. An extremum can be either global or local. In my problem, most of my interest is in finding the local minimum of the energy function. Some numerical methods require the derivatives of the functions, some don't. I will choose between these two kinds of methods.

4.1 One-Dimensional Function

This is the simplest case. For one-dimensional minimization, we don't need to calculate the derivative, which could be expensive for some complex functions. We can use the "Bracket" method to bracket a minimum.

- Bracket Method (Golden Section Search)

This method is very similar to the bisection method finding roots of functions in one dimension. Instead of bisecting the interval, we need a triplet of points, say $a < b < c$, such that $f(b)$ is less than both $f(a)$ and $f(c)$. We know that the function has a minimum in the interval (a, c) . The strategy is:

- Choose a new point x between b and c (or between a and b).
- If $f(b) < f(x)$

Then

Take the new bracketing triplet of points is (a,b,x)

Else

Take the new bracketing triplet of points is (b,x,c)

- (c) Repeat (a) through (b) until some stopping criterion is met (length of the interval is less than tol)

Question is how to choose the new point x , given (a,b,c) . Without the proof, I will give one of my strategy that is to pick up the middle point a fractional distance 0.38197 from one end, and 0.61803 from the other end. These fractions are called golden section. Another choice is to use inverse parabolic interpolation to determine the new point x . Basic idea is to take the minimum of the parabola to be the new point.

By the way, initially we should determine a triplet that brackets a minimum of the function.

4.2 Multidimensional Function

4.2.1 Method with the computation of first derivatives

The basic idea of the iterative descent method of optimization is

Given multidimensional function $f(x)$.

- (a) Choose an initial approximation $x^{(0)} \in \Omega$
- (b) Given $x^{(k)}$, calculate a direction $p^{(k)}$
- (c) Find $\alpha^{(k)} > 0$, such that $f(x^{(k)} - \alpha^{(k)} p^{(k)}) < f(x^{(k)})$ (Linear search)
- (d) Set $x^{(k+1)} = x^{(k)} - \alpha^{(k)} p^{(k)}$
- (e) Repeat (b) through (d) until some stopping criterion is met.

One problem in this option is how to get the derivatives of the original function. I once used ADIC ([Automatic Differentiation Tool for ANSI-C](#)) that is a tool for the automatic differentiation of programs written in ANSI C to generate the derivative functions of the original function, but later I found that those machine generated code is far beyond accurate and efficient. I started to find the derivative of the energy function using the basis functions of each bilinear element.

- Conjugate Gradient Method

Now I consider the case where I have not just a function $f(x)$, but also the gradient $\nabla f(x)$.

We may have different choices of the searching direction. The simplest idea is the Steepest Descent Method. Start at the point P_0 . As many times as needed, move from point P_i to the point P_{i+1} by minimizing along the line from P_i in the direction of the local downhill gradient $-\nabla f(P_i)$. Although it is a very easy way to find the descent direction, but obviously we can find it is very inefficient, because this direction is not pointing directly to the minimum point in most cases.

We know the conjugate gradient method for the linear system. It can also be used in minimizing a function.

(a) Starting with an arbitrary initial vector x_0

For $i = 0, 1, 2, \dots$ do (b) – (f)

(b) Set $g_i = \nabla f(x_i)$, $\beta_i = \frac{(g_i - g_{i-1}, g_i)}{\|g_{i-1}\|^2}$, $\beta_0 = 0$ (Here we use Polak-Kibière variation)

(c) Take the searching direction $h_i = g_i + \beta_i h_{i-1}$

(d) Line search for α_i , such that $f(x_i - \alpha_i h_i) < f(x_i)$

(e) Set $x_{i+1} = x_i - \alpha_i h_i$

(f) Exit when stopping criterion is met.

Remark: 1). In step (b), another choice of β_i is $\beta_i = \frac{\|g_i\|^2}{\|g_{i-1}\|^2}$, which is Fletcher-Reeves variation.

2). In step (d), we can construct a quadratic approximation of the one-dimension function $f(\alpha) = f(x_i - \alpha h_i)$, then take α_i to be the value of α which minimizes the quadratic approximation.

3). The vectors satisfy the orthogonality and conjugacy conditions

$$(g_i, g_j) = 0; (h_i, h_j) = 0; (g_i, h_j) = 0$$

4). It can be proved that, for the nonconvex functional in 2.2, this conjugate gradient method can be guaranteed to converge to a local minimum.

4.2.2 Method without the computation of first derivatives

- Downhill Simplex method (Nelder-Mead's minimization method)

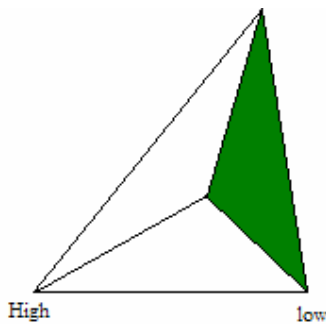
This is a method that is independent to the one-dimension minimization. The most promising benefit of this method is we don't need to evaluate the first derivative of the original function, which is very expensive in most cases.

In N dimensions, a simplex is the geometrical figure consisting of $N+1$ points and all their interconnecting line segments, polygonal faces. For instance, in two dimensions, a simplex is a triangle; in three dimensions it is a tetrahedron.

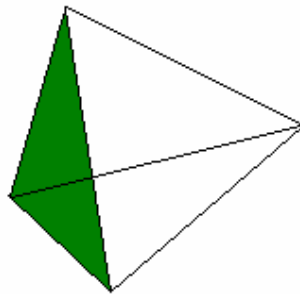
For the N -dimension problem, first we can give a starting guess, an N -vector of independent variables as the first point to try. The algorithm is then supposed to go downhill through the unimaginable complexity of an N -dimensional topography, until it encounters a minimum (local or global). Obviously the initial guess of the simplex method is $N+1$ points that construct an initial simplex. Then we would like to take some steps to move one vertex of this simplex, reflecting it around its current position, until we have enclosed the minimizer in the simplex. The next step is to shrink the size of the simplex to hone in on the minimizer.

The basic steps the downhill simplex method usually takes (demos in 2-D):

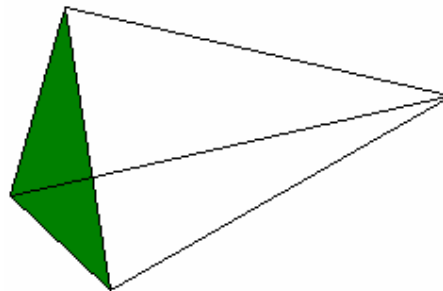
(a) Original simplex



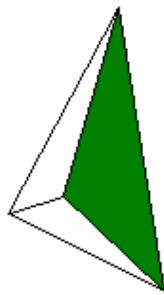
(b) Reflection



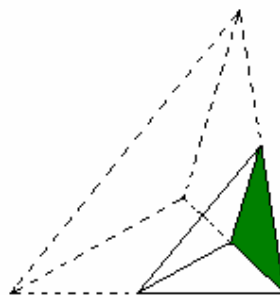
(c) Reflection and expansion



(d) Contraction



(e) Multiple contraction



I just started to learn this method, and I have the C-code that realizes this method from Numerical Recipe. Another good tool of this method is Matlab which has an implementation of this.

Although I can avoid the derivative of the energy function using this simplex method, the problem is there is no good convergence proof of this method. I don't know whether this method can converge to a local minimizer for my density function. I think it is worth trying.

5. Some Preliminary Work

In the past two months, I mainly focused on learning some basic concepts in material science, and getting familiar with the numerical methods of nonlinear optimization. I read some example codes in Numerical Recipe, and I also ran my test programs on matlab. The optimization toolbox in matlab helps me a lot.

I chose a Two-Well problem with the energy density

$$\Phi(C) = \kappa_1 (\text{tr}C - 2)^2 + \kappa_2 C_{12}^2 + \kappa_3 \left(\left(\frac{C_{11} - C_{12}}{2} \right)^2 - \varepsilon^2 \right)^2$$

Theoretically, the material with this energy density should have the *twinning* in crystal structure, which represents two different deformations. What I am doing is to simulate this microstructure numerically.

I am using the Finite Element Method, so I have set up the discretization format for this problem.

I finished some codes. One is the Configuration class that sets up the domain, boundary condition, discretized energy density, and the first derivative function of the energy function. I am working on the optimization module that is specifically designed for this energy function. I wrote the function of Conjugate Gradient method, but till now it only works with 1-D case. For the multidimensional case, my program doesn't converge. I am still debugging it.

I am also working on another option of the optimization method, Nelder-Meade method. I transformed my C code to matlab, and use the implementation of Nelder-Meade method in matlab to minimize my energy function. Unfortunately, till now, I don't have any good results can show.

Another task is to settle on a proper computer to do programming. Currently I am working on the SUN machine in math department. After long time waiting, I eventually got an account in the Linux cluster in UMIACS. I have transformed everything to the Linux cluster, but part of my programs still need to be adapted for the Linux machine.

I am using CVS system on math server to manage my codes, and I will also use CVS in the Linux cluster.

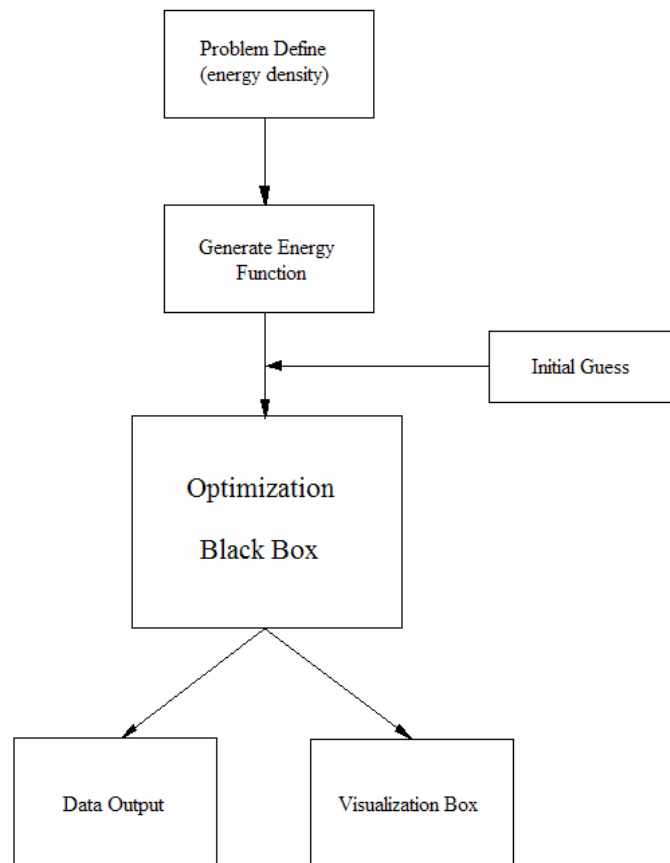
Currently, all the codes are written by myself. Initially I linked the SUN libraries, but later I transformed to the Linux cluster. I rewrote all those basis functions by myself to avoid the error. I will definitely use some packages to save time later.

6. Future Work

◆ During the winter break,

- 1). I need to finish the optimization framework based on Conjugate Gradient method for the two-well problem.
- 2). I will investigate the Nelder-Mead Method in detail, try to use this method to get some results, so that I can compare them with the results of my Conjugate Gradient method.
- 3). Settle down on the new Linux cluster. Get familiar with those libraries available in red cluster.
- 4). Explore the Parallel Computing for my project, so that I can run my program on the fine mesh to get more accurate solution.

Basically, the ideal structure of my optimization tool for the martensitic problem is like



◆ In next semester

1). Move to the energy problem with the surface energy, whose energy function in 1-D case has the form

$$E(u) = \int_0^1 [\phi(u') + u^2 + \varepsilon(u'')^2] dx$$

where $\phi(f) = (f - 1)^2 (f + 1)^2$

We notice that it has the second derivative component that could bring us extra trouble in optimization.

- 2). Explore the adaptive finite element method.
- 3). Test my program for different boundary condition.
- 4). Complete the Visualization class that shows the my simulation results.

References

- [1] Charles Collins, *Computing Microstructures: The problems and Some Solution*, Department of Mathematics, University of Tennessee, (1997)
- [2] Martin Kruzik, *Numerical Approach to Double Well Problems*, Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, Pod vodarenskou vez 4, CZ-182 08 Praha 8 (1998)
- [3] Kaushik Bhattacharya, *Theory of Martensitic Microstructure and The shape-memory Effect*, Division of Engineering & Technology, California Institute of Technology, (1997)
- [4] Charles Collins, *Computational of Twinning*, Department of Mathematics, University of Michigan, Ann Arbor, MI, (1991)
- [5] Bo Li, *Theory and Computation of Martensitic Microstructure*, Department of Mathematics, University of Maryland at College Park, (1999)
- [6] Bo Li, *Analysis and computation of martensitic microstructure*, Ph.D Thesis, University of Minnesota, (1996)
- [7] Bo Li and Mitchell Luskin, *Nonconforming finite element approximation of crystalline microstructure*, Math. Comp., 67:223, 917-946, (1998)