

# Computational Control of a Crystallization Process

John Burkardt  
Mathematics Department  
Virginia Tech, Blacksburg, VA

09 April 1996

## 1 Introduction

The silicon chips that form the heart of an electronic computer are expensive, and yet they are manufactured from sand, an extremely cheap material. The price of a finished silicon chip reflects both the cost of manufacturing the blank chip, and the cost of etching the desired circuits on that blank.

The manufacturing process for blank silicon chips transforms the raw sand, or silicon dioxide ( $SiO_2$ ), into lumps of polycrystalline silicon, and then produces a large cylindrical crystal. This crystal is sliced into thin wafers out of which the blank silicon chips are stamped.

The creation of the cylindrical crystal is very expensive, requiring as much as a day, and \$1,000,000 of electrical energy. While the original process was unreliable and little understood, the crystallization is now done automatically, and with very high reliability.

Recently, attempts have been made to construct devices, and even general purpose computers, that are based on the transmission of photons rather than electrons. Such “optical-based” devices cannot be constructed from silicon, but rather from more exotic compounds for which there is comparatively little engineering and fabricating experience.

One compound which has shown promise for optical computing is the semiconducting compound Indium Phosphate ( $InP$ ). If the Czochralski process is applied to the manufacture of suitable ( $InP$ ) crystals, however, significant problems arise. The first problem is that, instead of dealing with the stable element silicon, we have to combine Indium with Phosphorus, an extremely reactive, and in fact explosive, chemical. For efficiency, this reaction must be carried out at a pressure of five atmospheres, which means that the open crucible used for silicon must be replaced by a pressurized chamber into which the two reagents are introduced. The rate at which the reagents enter the chamber must be carefully controlled, so that all the phosphorus reacts with the Indium, leaving no dangerously reactive residue. If the compound is properly formed, then heat may be added to melt the compound and begin the process of crystallization. The process of crystallization must be closely monitored, but this is hindered by the fact that the process takes place in a closed chamber. The fabricator must observe the forming of the crystal through a special probe of limited scope; the probe itself is liable to obstruction during the formation of the ( $InP$ ) compound. This process is currently extremely unreliable. As often as one time in four, the entire apparatus explodes when the crucible is opened at the end, because a residue of excess phosphorus reacts with atmospheric oxygen. Moreover, the crystals produced by this method are not as satisfactory, having many flaws that make portions of the crystal unsuitable.

Thus, it seems useful to develop a computer model of the crystallization process that can help to explain current results, predict the effect of small changes in the process, and suggest the best ways to reduce the cost and increase the reliability.

## 2 The Czochralski Process for Silicon Crystallization

The current standard method for producing silicon chips is the *Czochralski Process*. This method begins with some amount of heterogeneous silicon, which can be thought of as a pile of very pure sand. This disordered mass is to be transformed into a cylindrical monocrystal of silicon roughly the size of a thermos bottle.

The raw silicon, called the “charge”, is placed in a quartz crucible, which is rotated at a rate  $\omega_1$ , which is usually constant, and which we will think of as being clockwise. The temperature of the charge is raised either through radio-frequency energy, or thermal resistance, to silicon’s melting point of  $1415C$ , at which temperature the charge liquefies, and is termed the “melt”.

Above the crucible is a platform that is rotating in the opposite (counter-clockwise) direction. From this platform, a small seed crystal of silicon is lowered until it just touches the melt. The melted silicon that touches the crystal has a tendency to cool, solidify, and attach itself to the crystal, extending the crystal’s regular structure, and, at least initially, increasing the area of contact between the crystal and melt. Gradually, the crystal extends over a large portion of the melt surface. Because of the rotation, the edge of the growing crystal area is very close to being circular, and the crystal itself is roughly the shape of a short, wide disk. The size of this disk can be monitored by eye, or by a laser device. When the disk has reached the desired diameter, the crystal is very slowly drawn upward. If the drawing rate is properly controlled, more melted silicon crystallizes along the bottom of the disk just as fast as it is drawn upward, so that cylindrical crystal begins to form. The drawing process is continued until almost all of the silicon has crystallized. Finally, the crystal is drawn up more rapidly, causing the crystal to taper off with a rounded bottom.

### 3 Efficiency and Costs of the Czochralski Process

The Czochralski process can fail in a number of ways. Gross mistakes in temperature control will mean that the solid silicon never melts, or never crystallizes once it melts, or forms crystals on the crucible wall. It can even happen that when the seed crystal is introduced, it melts away.

Even if the overall temperature is properly controlled, there may be local fluctuations in heat. A local “hot spot” may remelt portions of the incipient crystal. A “cool spot” may cause crystallization to occur too quickly, resulting in twinning, other crystal defects, or even a distortion of the desired gross cylindrical shape. The rotations of the crucible and the seed crystal are primarily intended to reduce the effects of hot spots.

The drawing rate, that is, the speed with which the growing crystal is pulled out of the melt, must be carefully controlled as well. At the beginning and end of the process, care must be taken to properly form tapered boundaries. During the rest of the process, the drawing rate must stay within a narrow range of suitable speeds, or the crystal will deviate from the desired cylindrical shape. The drawing rate can also affect the generation of crystal defects.

The Czochralski process for silicon crystallization has been refined to the point where large crystals are regularly created with few flaws. Now that crystals can be formed reliably, it is important to determine how to create them *cheaply*. The primary expense in the process is the enormous quantities of electrical power needed to melt the silicon, and maintain it in a molten state. The growth of a typical crystal requires 24 hours, and electrical energy worth \$1,000,000.

### 4 The Idealized Geometry

In order to apply computational techniques to our problem, we begin by making a simplified sketch of the region. Our first simplification is fairly drastic; we assume *axisymmetry*, that is, that the region is completely symmetric about some axis of symmetry. If we accept this approximation, then we can understand everything about our problem by looking at a two dimensional slice of the region that passes through the axis of symmetry, and in fact, we only need half of that slice.

In our particular sketch, the region will be shown lying on its side, so that the axis of symmetry, and the direction we would normally think of as “up” is the horizontal  $X$  axis. This slightly confusing choice is only made for the convenience of programming.

Our preliminary sketch of the idealized region is Figure 1. The region is bounded by an axis of symmetry, the crucible, and the free surface. In this simple model, the bottom of the crystal is flat, and does not extend below the plane defined by the free surface.

Our model of the geometry retains the important features of the real problem, but the simplifications that we have made will allow us to apply computational methods that will efficiently predict what happens.

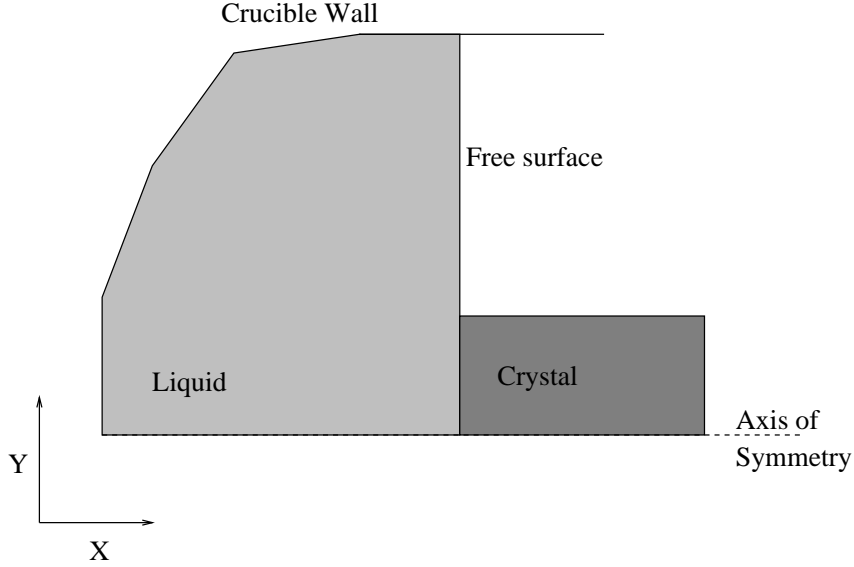


Figure 1: A simplified model of the physical region.  
The crucible is halved, and lying on its side.

## 5 Discretizing the Geometry

We wish to replace Figure 1 by an equivalent picture in a more conventional presentation, standing up instead of lying down, as in Figure 2. And while Figure 1 was essentially drawn by hand, our new figure has been drawn by a program which we will be using to display the results of our computation.

Having settled on our idealized geometry, we must now consider how to describe and analyze it. In order to make the problem computationally feasible, we must “discretize” the geometry; that is, instead of seeking the behavior of the solution functions at every point in the region, we will only seek such values at a finite set of points, which must be chosen in a representative way. To do so, the computational region will be divided up into “control volumes”, and from each control volume a single point will be chosen at which the values of the state variables will be sought. The control volumes will be quadrilateral, and hence may be defined by specifying their vertices, which will be called the “corner nodes”. For ease of programming, the corner nodes will be arranged in a rough rectangle of  $L$  rows and  $M$  columns, with corner node (1,1) in the lower left corner, node (L,1) in the upper left corner, and so on. A certain amount of bending and stretching will be allowed in the rows and columns, so that the shapes of boundaries and interfaces may be closely followed.

We will endow this region with an  $(x, y)$  coordinate system. The index of the corner node in row  $I$  and column  $J$  will be denoted by  $[I, J]$ , and its actual  $(x, y)$  coordinates will assume to be stored in arrays  $XC$  and  $YC$ , so that its position may be denoted by  $(XC_{i,j}, YC_{i,j})$ .

Again, for reasons that will become clear later, we will consider a “reference space”, with the following properties:

- the coordinate system is denoted by  $(\xi, \eta)$ ;
- corner node  $[I, J]$  has  $(\xi, \eta)$  coordinates  $\xi = I, \eta = J$ ;
- the data arrays  $XC$  and  $YC$  can be regarded as discretizations of maps from the (discretized) reference space to the (discretized) physical space, which we may be able to extend in a natural way.

For convenience, we will arrange these nodes in a “logical rectangle”

Each control volume can be completely defined by specifying the location of its “corner nodes”. Thus, we can imagine the entire set of control volumes to be specified by two arrays,  $XC_{i,j}$  and  $YC_{i,j}$ , with  $I$  and  $J$  ranging from 1 to  $L$  and  $M$  respectively, and  $XC$  and  $YC$  being the coordinates of the nodes in the physical

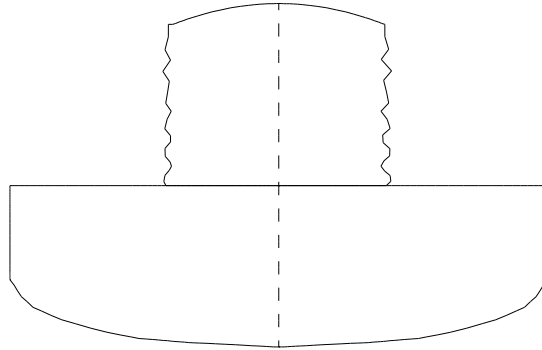


Figure 2: The computational region.  
 The crucible is standing upright,  
 and the “missing half” is shown as well.

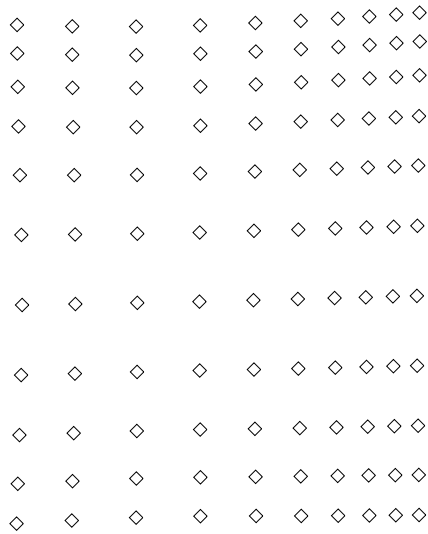


Figure 3: The corner nodes.  
 The rectangular arrangement is evident,  
 but also the variable spacing, and the deviations from straightness.

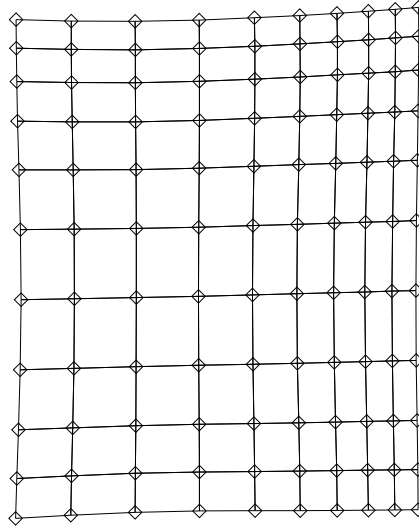


Figure 4: The control volumes defined by the corner nodes.

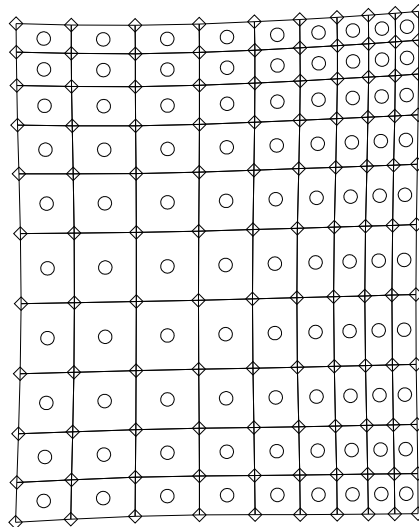


Figure 5: The primary nodes defined by the control volumes. Each primary node is the average of the four adjacent corner nodes.

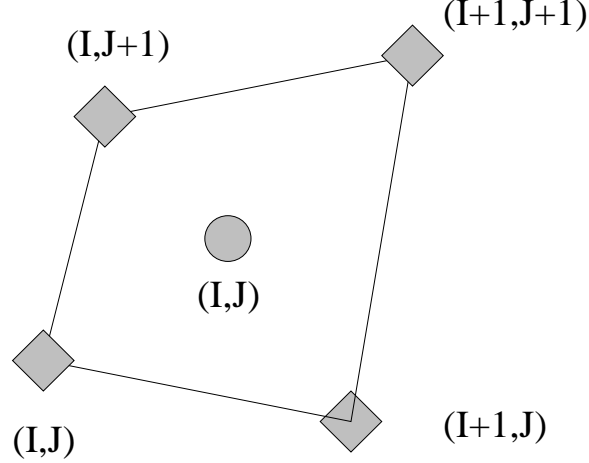


Figure 6: A single control volume,  $CV_{i,j}$ .  
The points  $(XC, YC)$  are shown as diamonds;  
The primary point  $(XP, YP)$  is shown as the central circle.

space. These corner nodes will define  $(L - 1) \times (M - 1)$  control volumes, with control volume  $CV_{i,j}$  defined by the corner nodes whose indices are  $(I, J)$ ,  $(I + 1, J)$ ,  $(I, J + 1)$  and  $(I + 1, J + 1)$ .

At the center of each control volume, defined by the average of the four vertices, we define a second type of point, called a “primary node”. We think of a primary node as being the representative point in a given control volume. In other words, when we wish to speak of the temperature in a control volume, we will refer to the temperature at the primary node. In this way, the values of the physical quantities at a discrete set of points will be allowed to represent what in real life are continuously varying quantities. We may designate the coordinates of a primary node by  $XP_{i,j}$  and  $YP_{i,j}$ , where we assume that primary node  $(I, J)$  lies inside of control volume  $CV_{i,j}$ . Note that the coordinates of the primary nodes are simply the average of the coordinates of the corner nodes. In particular,

$$XP_{i,j} = \frac{1}{4}(XC_{i,j} + XC_{i+1,j} + XC_{i,j+1} + XC_{i+1,j+1}). \quad (1)$$

A typical control volume is shown in Figure 6.

Since every set of four “contiguous” vertices defines a control volume, if we make sure that the first and last columns and rows of control volumes lie on the extremities of the region, then we have effectively broken up the entire region into a collection of control volumes, which interact with each other and the outside world through the boundary lines defined by pairs of vertices, and which have values of physical quantities associated with the central primary node.

In Figure 7, we display the dissection of a portion of the computational region into control volumes. (This portion includes the point where the crystal, liquid, and free surface meet. Compare with Figure 2.) The corner nodes are marked as diamonds, and the primary nodes as circles. The diagram should make clear how the corner nodes are arranged in rows and columns that may bend, how the control volumes are quadrilaterals that may be distorted, and how the primary nodes lie at the centers of the control volumes.

We will see later that at each primary node we will apply the state equations in order to determine the local values of the state variables. In the physical problem, however, the state variables inside the region cannot be determined without specifying certain values along the boundary of the region. The same fact applies for our discretized problem; in order to find the state values at primary nodes inside the region, we are going to need to determine values along the boundary. Hence, we must include a strip of “control areas”, that is, one dimensional control volumes, along the boundaries of the region, and place primary nodes in the centers (or better, the midpoints) of these areas. Figure 8 shows the situation that obtains for the lower left hand corner of the computational region. The extra primary nodes that have been added to handle boundary conditions appear along the left and lower boundaries of the picture.

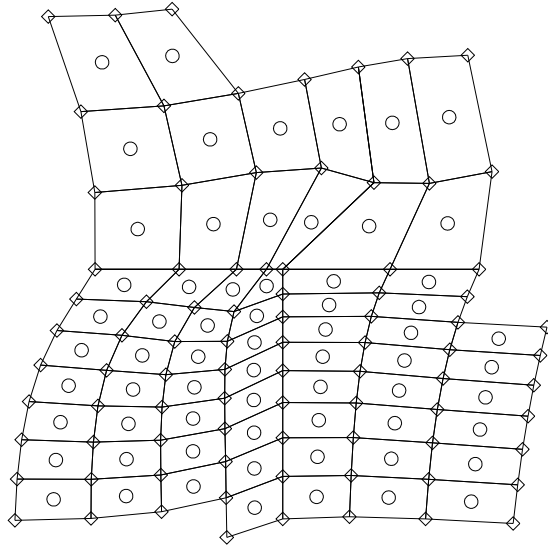


Figure 7: Control volumes in a portion of the computational region.

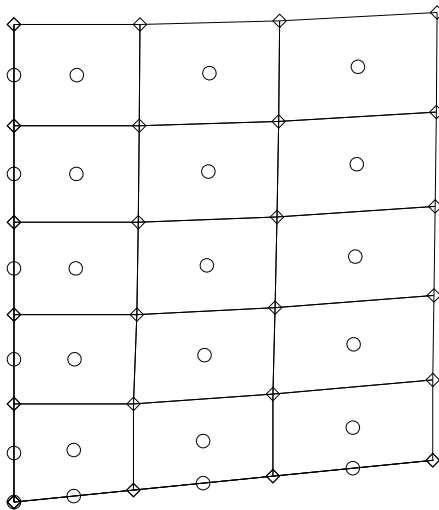


Figure 8: Primary nodes are added along boundaries.  
The lower left corner of the region is shown.

To keep things consistent, we will simply repeat the  $XC$  and  $YC$  values along the outer edges of the region, which will have the effect of automatically creating these control areas as part of the creation of the control volumes, without requiring us to store them in any special way. Note that a primary node will also be generated at each corner of the region.

Finally, we should note that the idealized geometry of Figure 1 may include curved surfaces, while the computational geometry of Figures 2 and 7 is built entirely from quadrilaterals with straight sides. Thus we may expect some small error in our calculations simply because we do not properly match curved boundaries. Such errors can be reduced by decreasing the size of the control volumes near such boundaries, at the expense of greater computational effort.

## 6 Discretizing the Physical Equations

We have now discretized the original physical region into a computational region, which is made up of a collection of control volumes. At the center of each control volume is a primary node. We have already suggested that, for each physical quantity of interest, such as temperature, we will only try to compute values at primary nodes. Thus, the original problem of determining the temperature function  $T(X, Y)$  everywhere is replaced by the discrete problem of determining the collection of temperature values  $T_{i,j} \equiv T(XP_{i,j}, YP_{i,j})$ .

Now in the original physical situation, the state functions such as the temperature  $T(X, Y)$  are assumed to satisfy a collection of equations comprising:

- state equations, generally differential equations which apply at points  $(X, Y)$  in the interior of the region, and for all time  $T$ , except for the initial time  $T_0$ , and except at internal interfaces. These equations may depend on the local phase of the material;
- initial conditions, which specify or constrain the state functions throughout the region at the initial time;
- boundary conditions, which specify or constrain the state functions along the physical boundaries of the region;
- interface conditions, which specify how the region is broken down into subregions of different phases, how these subregions may change with the physical state, and how state variables may change across such interfaces.

This system of equations is called *continuous*, in the sense that the physical properties such as temperature are assumed to be defined over a smooth continuum of spatial points  $(X, Y)$  and time, and to be continuously differentiable with respect to  $X$ ,  $Y$ , and  $T$  to as many orders as necessary.

For our discrete model, we are going to have to replace this system of equations by a discrete system. By confining our interest in the state functions to their values at the primary nodes, we have essentially replaced the continuous state by discrete tables of data. The differential operations applied in the continuous state equations cannot be directly applied to such discrete tables of data. In order to produce our discrete set of state equations, we must decide where they will apply, and how we are going to approximate terms involving spatial or temporal derivatives.

We expect the state equations, and possibly some of the related conditions, to be differential, that is, to describe how the state variables change over very small distances. In our computational model, however, we only have state variable values at fixed, and relatively large, distances.

## 7 Setting Up the Reference Space

In fact, it is convenient to endow the “logical” space with a simple coordinate system,  $\xi$  and  $\eta$ , so that we may consider the correspondence between the table of indices  $(I, J)$  and the location of the physical nodes  $(XC_{i,j}, YC_{i,j})$  to be a mapping from an undeformed reference space to the physical space. The idea of using an undeformed reference space recalls the use of a master reference element in finite element applications. It is introduced for a similar reason: the need to compute spatial derivatives on a distorted mesh.

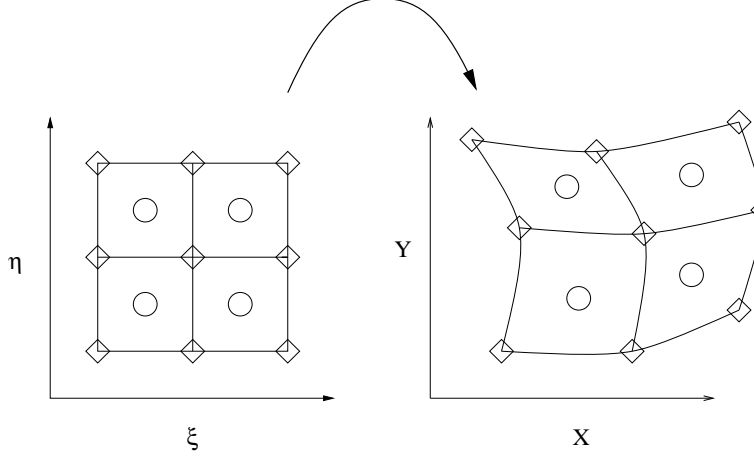


Figure 9: The mapping from  $(\xi, \eta)$  to  $(X, Y)$ .

## 8 Computation of Spatial Derivatives

In order to set up the discretized state equations at any interior primary node, we need to define certain coefficients and right hand side quantities that will be based on known or previously computed data. This data includes the values and spatial derivatives of various state functions. We may assume that we always have available a previous approximation of the state function values; the initial conditions give us this information at the first time step, and the solution of the discretized state equations gives us the data we need for the next time step.

But although we always have available the values of the state functions at the primary nodes, we must somehow compute spatial derivatives with respect to the physical directions  $X$  and  $Y$ , using data that does not lie along  $X$  and  $Y$  coordinate lines. To do so, we will regard the data as originally lying in an undistorted reference space, where the derivatives are easy to compute. Then we consider how the reference space is mapped to the physical space, in order to transform the reference derivatives into useful physical derivatives.

Recall that we have two sets of nodes that we used to divide up the physical space: corner nodes, which defined the corners of control volumes, and primary nodes, which lay at the center of each control volume. While the value of a typical state variable  $\Psi$  will be computed at the primary nodes, spatial derivatives such as  $\frac{\partial \Psi}{\partial X}$  will need to be evaluated on the midpoint of a wall of the control volume.

We will construct our reference space specifically in order to make the calculation of these spatial derivatives as simple as possible. Therefore, we will assume that the mapping of the reference space  $(\xi, \eta)$  to the physical space  $(X, Y)$  has the property that the pre-images of the mid-side nodes form a regularly spaced array of points that lie along  $\xi$  and  $\eta$  coordinate lines. We will even go further, and assume that the spacing between these pre-images is 1 in both  $\xi$  and  $\eta$  directions.

Now we shall find that we need to estimate the value of  $\Psi$  at the mid-side nodes. To do so is fairly simple. We know that the mid-side node lies on a straight line between two primary nodes, and it seems reasonable to take its value as a weighted average of the values at the primary nodes. The weight we will use is the relative physical lengths of the line segments connecting the two primary nodes to the mid-side node. Let us suppose we want to estimate the value of  $\Psi$  at a midside node  $(XN_{i,j}, YN_{i,j})$  which lies north of the primary node  $(XP_{i,j}, YP_{i,j})$  and hence south of  $(XP_{i+1,j}, YP_{i+1,j})$ , then we first compute the lengths:

$$d_1 = \sqrt{(XP_{i,j} - XN_{i,j})^2 + (YP_{i,j} - YN_{i,j})^2} \quad (2)$$

$$d_2 = \sqrt{(XP_{i+1,j} - XN_{i,j})^2 + (YP_{i+1,j} - YN_{i,j})^2} \quad (3)$$

$$\alpha = \frac{d_2}{d_1 + d_2} \quad (4)$$

$$\Psi(XN_{i,j}, YN_{i,j}) = \alpha\Psi(XP_{i,j}, YP_{i,j}) + (1 - \alpha)\Psi(XP_{i+1,j}, YP_{i+1,j}) \quad (5)$$

The above equations still hold if the adjacent primary node is actually a boundary node. However, in that case, the adjacent primary node is coincident with the midside node, and so the midside value is easily computed. Note that every interior primary node always has a primary node neighbor in all four directions, and that we generally do not need to worry about computing spatial derivatives at boundary primary nodes, since we will not be defining state equations there.

Our next step, then, is to define a “reference function”. Assuming we are interested in some quantity  $\Psi(X, Y)$ , let us suppose that we have obtained a sample value of this quantity at each of our primary nodes  $(XP_{i,j}, YP_{i,j})$ . Then we may consider the set of “preimages” of the primary nodes in  $(\xi, \eta)$  space, and at each preimage  $(\xi_i, \eta_j)$ , define the corresponding reference function  $\psi(\xi, \eta)$  with the property that:

$$\psi(\xi_i, \eta_j) \equiv \Psi(X(\xi_i, \eta_j), Y(\xi_i, \eta_j)) \quad (6)$$

$$= \Psi(XP_{i,j}, YP_{i,j}). \quad (7)$$

In the reference space, the data is sampled along coordinate lines, which is exactly what we want. Hence, it is easy to estimate partial derivatives of the reference function  $\psi$  with respect to the reference coordinates:

$$\frac{\partial\psi(\xi, \eta)}{\partial\xi} \approx \frac{\psi(\xi + \Delta\xi, \eta) - \psi(\xi, \eta)}{\Delta\xi} \quad (8)$$

and we will assume that in our table, the spacing between successive points represents a sampling with  $\Delta\xi = \Delta\eta = 1$ , so that in fact, at any reference node  $(\xi_i, \eta_j)$ , we may write:

$$\frac{\partial\psi(\xi_i, \eta_j)}{\partial\xi} \approx \psi(\xi_{i+1}, \eta_j) - \psi(\xi_i, \eta_j) \quad (9)$$

Now to compute derivatives with respect to the physical direction  $X$ , we assume that our transformation  $(\xi, \eta) \rightarrow (X, Y)$  is continuously differentiable. Therefore, we can estimate a “transformation derivative” such as  $\frac{\partial X(\xi, \eta)}{\partial\xi}$  by

$$\frac{\partial X(\xi, \eta)}{\partial\xi} \approx \frac{X(\xi + \Delta\xi, \eta) - X(\xi, \eta)}{\Delta\xi} \quad (10)$$

Thus we can estimate the *Jacobian matrix*:

$$J(\xi, \eta) = \begin{pmatrix} \frac{\partial X(\xi, \eta)}{\partial\xi} & \frac{\partial X(\xi, \eta)}{\partial\eta} \\ \frac{\partial Y(\xi, \eta)}{\partial\xi} & \frac{\partial Y(\xi, \eta)}{\partial\eta} \end{pmatrix} \quad (11)$$

If we now use standard techniques to invert the Jacobian matrix, we arrive at the inverse Jacobian matrix, containing the “inverse transformation derivatives”:

$$J^{-1}(X, Y) = \begin{pmatrix} \frac{\partial\xi(X, Y)}{\partial X} & \frac{\partial\xi(X, Y)}{\partial Y} \\ \frac{\partial\eta(X, Y)}{\partial X} & \frac{\partial\eta(X, Y)}{\partial Y} \end{pmatrix} \quad (12)$$

Now we have all the tools we need to compute a “physical derivative” such as  $\frac{\partial\Psi(X, Y)}{\partial X}$ .

Then we use the chain rule to write:

$$\frac{\partial\Psi(X, Y)}{\partial X} = \frac{\partial\psi(\xi(X, Y), \eta(X, Y))}{\partial\xi} \frac{\partial\xi(X, Y)}{\partial X} + \frac{\partial\psi(\xi(X, Y), \eta(X, Y))}{\partial\eta} \frac{\partial\eta(X, Y)}{\partial X}. \quad (13)$$

Now we wish the approximate spatial derivative resulting from Equation 13 to be associated with the primary node  $(XP_{i,j}, YP_{i,j})$ .

Therefore, we first use versions of Equation 5 to estimate  $\Psi$  at the four midside nodes  $(XN_{i,j}, YN_{i,j})$ ,  $(XS_{i,j}, YS_{i,j})$ ,  $(XE_{i,j}, YE_{i,j})$  and  $(XW_{i,j}, YW_{i,j})$ . If this data is then used in Equation 8, we can estimate  $\frac{\partial\psi(\xi(XP_{i,j}, YP_{i,j}), \eta(XP_{i,j}, YP_{i,j}))}{\partial\xi}$  and  $\frac{\partial\psi(\xi(XP_{i,j}, YP_{i,j}), \eta(XP_{i,j}, YP_{i,j}))}{\partial\eta}$ .

The other terms in Equation 5 come from the the inverse Jacobian. In order that they also be associated with the primary node, it is only necessary that the estimation of terms like  $\frac{\partial X(\xi, \eta)}{\partial \xi}$  should be done by comparing coordinates at the appropriate opposite midside nodes:

$$\frac{\partial X(\xi(XP_{i,j}, YP_{i,j}), \eta(XP_{i,j}, YP_{i,j}))}{\partial \xi} \approx XN_{i,j} - XS_{i,j}. \quad (14)$$

Hence we have seen that if we are given the values of the state variables at the primary nodes and the coordinates of the corner nodes of the control volumes (which in turn define the midside nodes and primary nodes), we can estimate the spatial derivatives of the state variables at those same nodes.

## 9 Defining Cost Functionals

We will soon be considering a variety of solutions of the state equations, and we will want to find the “best” one. This means we must have some quality in mind that is desirable, so that for any solution we can measure the amount of this quality, and rate it relative to other solutions. This quantity is often called a “cost functional”. The definition of this cost functional is entirely up to us. We must be guided by experimental knowledge; in the case of crystallization, for instance, the effect of various patterns of the state variables on the regularity of the crystal is known or suspected. For instance, great vorticity, high turbulence, or a steep heat gradient are known to increase the likelihood of a bad crystal.

Practical experience has shown that strong flow fields in the melted crystal can cause crystal irregularities. Flaws in the crystal can only be detected at the end of the crystal growth process. But the flow field is directly observable experimentally, and can be reliably determined computationally. This suggests that one way to improve crystal quality would be to control the flow magnitude. For a two dimensional flow, this quantity is represented by  $(u^2(x, y, t) + v^2(x, y, t))^{\frac{1}{2}}$ . Since we are interested in controlling this quantity over the entire flow region, and a specific range in time, we square the pointwise quantity and integrate:

$$J^{UV}(u, v) = \int_{t_0}^{t_1} \int_{\Omega} (u^2(x, y, t) + v^2(x, y, t)) \, dx \, dy \, dt. \quad (15)$$

This quantity is our first example of a cost functional, that is, a formula which takes quantities associated with a solution to the state equations and returns a scalar value, which it is our desire to minimize. We assume that the time interval and flow region are given, and that  $u$  and  $v$  are solutions of the given state equations.

For later use, we now compute what we may think of as the partial derivative of  $J^{UV}(u, v)$  with respect to  $u$ , or, more properly, the Frechet derivative or first variation. Since our argument  $u$  is actually a function, we must be somewhat careful in our reasoning. We suppose that  $u$  is incremented by a scalar multiple  $\epsilon$  of some finite perturbation function  $w$ . Then, if the limit exists, we can define the partial derivative with respect to  $u$ , operating on  $w$ , as:

$$J_u^{UV}(u, v)w = \lim_{\epsilon \rightarrow 0} \frac{J^{UV}(u + \epsilon w, v) - J^{UV}(u, v)}{\epsilon} \quad (16)$$

If we gather the integrands on the right hand side and simplify, we get:

$$J_u^{UV}(u, v)w = \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} \int_{t_0}^{t_1} \int_{\Omega} (2\epsilon w u + \epsilon^2 w^2) \, dx \, dy \, dt \quad (17)$$

$$= \int_{t_0}^{t_1} \int_{\Omega} (2 w(x, y, t) u(x, y, t)) \, dx \, dy \, dt \quad (18)$$

It is also possible that, instead of minimizing the flow magnitudes, we wish to keep them close to some particular desirable magnitude,  $V_{ave}$ . In that case, the cost function might have the form:

$$J^{VAVE}(u, v) = \int_{t_0}^{t_1} \int_{\Omega} (\sqrt{(u^2(x, y, t) + v^2(x, y, t))} - v)^2 \, dx \, dy \, dt. \quad (19)$$

Further, we may simply want to control the velocity magnitude for the flow cells under the crystal:

$$J^{UVCR}(u, v) = \int_{t_0}^{t_1} \int_{\Omega_{crystal}} (u^2(x, y, t) + v^2(x, y, t)) \, dx \, dy \, dt. \quad (20)$$

If, on the other hand, we wish to control the vorticity, we must first estimate the vorticity from the state variables:

$$Vort(x, y) = \frac{\partial V(x, y)}{\partial x} - \frac{\partial U(x, y)}{\partial y} \quad (21)$$

and then compute the cost functional as:

$$J^{Vort}(u, v) = \int_{t_0}^{t_1} \int_{\Omega} Vort(x, y, t)^2 \, dx \, dy \, dt. \quad (22)$$

We may also choose to minimize the deviation of the temperature in the liquid region from some desired temperature  $TF$ , which would suggest the cost functional

$$J^{TAVE}(u, v) = \int_{t_0}^{t_1} \int_{\Omega} (T(x, y, t) - TF)^2 \, dx \, dy \, dt. \quad (23)$$

Another suggestion has been to control the heat flux leaving the liquid through the free surface or into the crystal:

$$J^{HFLEX}(u, v) = \int_{t_0}^{t_1} \int_{LiquidBoundary} (k(x, y) \frac{\partial T(x, y, t)}{\partial n})^2 \, ds \, dt. \quad (24)$$

where  $k(x, y)$  is the thermal conductivity.

## 10 The Discretized Cost Functional

Now let us assume that we have made a model of the process. We have a two dimensional “computational region”, which is a schematic diagram of a slice of the Czochralski process. Throughout this region, we have arranged a grid of points. This grid is “logically” rectangular, that is, we can think of it as arranged in  $L$  rows and  $M$  columns, but the actual appearance of the points in the schematic may seem irregular.

These points will be called “primary nodes”. Any such point can be specified by giving its row and column number, which we will usually denote as  $I$  and  $J$ . At each primary node, we will expect to compute values of all the solution parameters; in particular, the values  $U(I, J)$  and  $V(I, J)$  of the horizontal and vertical velocity.

We will also divide up the space between these primary nodes, using a system of “secondary” or “corner” nodes, so that each primary node is at the center of a *control volume*. So long as the sides of the control volume are straight, we may compute its area using the standard formula for the area of a polygon that is bounded by vertices  $P_1, P_2, P_3, P_4$ , where the coordinates of  $P_i = (x_i, y_i)$ :

$$AREA(P_1, P_2, P_3, P_4) = \frac{1}{2}(x_1(y_2 - y_4) + x_2(y_3 - y_1) + x_3(y_4 - y_2) + x_4(y_1 - y_3)) \quad (25)$$

Since we already have indexed our nodes by coordinates  $(I, J)$ , we may rewrite this formula in those terms as:

$$\begin{aligned} AREA(I, J) = \frac{1}{2} & \left( XC_{i,j}(YC_{i+1,j} - YC_{i,j+1}) + XC_{i+1,j}(YC_{i+1,j+1} - YC_{i,j}) \right. \\ & \left. + XC_{i+1,j+1}(YC_{i,j+1} - YC_{i+1,j}) + XC_{i,j+1}(YC_{i,j} - YC_{i+1,j+1}) \right) \end{aligned} \quad (26)$$

and thus, we can compute the total computational area as:

$$AREA = \sum_{i=1}^L \sum_{j=1}^M AREA_{i,j} \quad (27)$$

If our physical region has curved boundaries, then the set of control volumes will not exactly coincide with the physical region and the area calculations will be only approximate; if this error is considerable, it can be reduced by a combination of increasing the number of control volumes, and decreasing the size of the control volumes near the boundary.

Once we have a way of measuring area, we can consider the approximation of integrals over the region. Since the area of a region is equal to the integral of the function  $f(x, y) = 1$  over that region, we may write

$$\begin{aligned} AREA(\Omega) &= \int_{\Omega} 1 \, d\Omega \\ &= \sum_{i=1}^L \sum_{j=1}^M \int_{\Omega_{i,j}} 1 \, d\Omega_{i,j} \end{aligned} \quad (28)$$

and then, by analogy, use the following approximation for the integral of any function:

$$\begin{aligned} \int_{\Omega} f(x, y) \, d\Omega &= \sum_{i=1}^L \sum_{j=1}^M \int_{\Omega_{i,j}} f(x, y) \, d\Omega_{i,j} \\ &\approx \sum_{i=1}^L \sum_{j=1}^M f(XP_{i,j}, YP_{i,j}) AREA_{i,j} \end{aligned} \quad (29)$$

This represents a simple Riemann sum estimate for the integral, whose accuracy is affected by how well the quadrilateral areas cover the possibly curved region  $\Omega$ , and by how rapidly the function  $f(x, y)$  varies over the region, as measured by the size of  $f_x$  and  $f_y$ . Both of these errors can be reduced by reducing the maximum size of the control volumes.

## 11 Solution Parameters

In a physical process such as the Czochralski method of creating silicon crystals, there are many factors which can be varied, and which influence the behavior of the state variables. These include the rate of rotation of the crucible, the shape of the crucible, the drawing rate, and the temperature at which the melted silicon is maintained. If the influence of such a quantity is to be specifically considered and analyzed, it is called a *solution parameter*.

A solution parameter represents a choice that is made when carrying out the given process. A parameter may be a single number, a geometric shape, or a complicated function of space and time.

Typical parameters for the Czochralski process include the rates of rotation for the crucible and the seed crystal, the drawing rate of the seed crystal, the shape of the crucible, and the temperature of the crucible.

We began with a model of a fixed physical process, whose state solution  $u(x, y, t)$  was completely determined. If we now regard certain quantities associated with the process as parameters that may vary, then only when we have made specific choices for each parameter will we get a state solution. Assuming the parameters are the only items that need to be specified in order to determine the state solution, then we may regard that state solution as an implicit function of the parameters. If we have a single parameter,  $\lambda$ , then we might write a typical state solution either as  $u(x, y, t; \lambda)$  or as  $u(\lambda)$ .

Depending on how a parameter  $\lambda$  has entered the problem, we may be able to show that solution components are actually differentiable with respect to the parameter. A partial derivative with respect to a problem parameter is often called the *sensitivity* with respect to that parameter, and is symbolized in the usual way as  $u_{\lambda}$ .

Generally, the dependence of  $u$  on  $\lambda$  is indirect, and therefore a quantity such as  $u_{\lambda}$  cannot be computed by direct differentiation of a formula for  $u$  involving  $\lambda$ , but rather by differentiation (or other analysis) of an implicit relationship between  $u$  and  $\lambda$ .

Since the state variables such as  $u$  may be regarded as functions of the parameters, it should be obvious that quantities based on the state variables can also be regarded in this way. Of particular concern is the cost function,  $J^{UV}(u, v)$ , which is defined for arbitrary  $u$  and  $v$ . If we assume, however, that  $u(\lambda)$  and  $v(\lambda)$

are the state variables that solve the parameterized state equations for a given value of  $\lambda$ , then we may regard the resulting cost as a function of  $\lambda$  as well:

$$\mathcal{J}^{UV}(\lambda) \equiv J^{UV}(u(\lambda), v(\lambda)). \quad (30)$$

Moreover, we may compute partial derivatives:

$$\mathcal{J}_\lambda^{UV}(\lambda) = J_u^{UV}(u(\lambda), v(\lambda))u_\lambda + J_v^{UV}(u(\lambda), v(\lambda))v_\lambda. \quad (31)$$

## 12 Minimization of a Parameterized Cost Functional

### References

- [1] Suhas Patankar, **Numerical Heat Transfer and Fluid Flow**, Hemisphere Publishing Corporation, 1980. QC320.P37 (I have a copy)
- [2] Fumio Shimura, **Semiconductor Silicon Crystal Technology**, Academic Press, 1989. TK7871.85 S523 (I have a copy)
- [3] Hui Zhang, "A Multizone Adaptive Grid Generation Technique for Simulations of Moving and Free Boundary Problems" PhD Dissertation, Polytechnic University at Brooklyn. (I have a copy)
- [4] Hui Zhang and M K Moallemi, "MAGG - A Multizone Adaptive Grid Generation Technique for Simulation of Moving and Free Boundary Problems," *Numerical Heat Transfer, Part B, Volume 27*, pages 255-276, 1995. QC319 N851 (I have a copy)
- [5] Hui Zhang and M K Moallemi, "Numerical Simulation of Hot-Dip Metallic Coating Process," *International Journal of Heat and Mass Transfer, Volume 38, Number 2*, pages 241-257, 1995. (I have a copy)
- [6] M K Moallemi and Hui Zhang, "A General Numerical Procedure for Multilayer Multistep IC Process Simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 13, Number 11*, pages 1379-1390, 1994. TK7874 I325 (I have a copy)
- [7] Hui Zhang and V Prasad, "A Multizone Adaptive Process Model for Crystal Growth at Low and High Pressures," *Journal of Crystal Growth* (in press), 1995. QD921 J6X. ??? ("Geology library?")
- [8] Hui Zhang, V Prasad, and M K Moallemi, "Application of MAGG - A Multizone Adaptive Generation Technique - for Simulations of Moving and Free Boundary problems," *Numerical Heat Mass Transfer, 1995. QC319.8 N851*