A Moving Finite Element Approach to Semiconductor Process Modelling in 1-D

John M. Hobbs

September 1993

Submitted to the

Department of Mathematics,

University of Reading,

in partial fulfillment of the requirements for the

Degree of Master of Science

The Moving Finite lement Method and modifications to the method are investigated for robustness and efficiency when applied to a problem in Semiconductor Process Modelling. The modifications carried out include the use of Penalty Functions, Gradient Weighting and Graph Massage, and the semiconductor modelling involves the prediction of the dopant concentration within the silicon that constitutes the semiconductor device, so that certain electrical properties of the device can be predicted.

I would like to thaneODDr.M.J.thaneODBananeODDr.PC]WpW.K.thanlIKOS))CHWwIeb)C

of the junction[1] is predicted accurately after diffusion is complete. The position of this junction occurs well into the tail region, which is some five or six orders of magnitude below the initial concentration, and this is a second source of numerical difficulty.

The aim of this dissertation is to formulate a simple model of the diffusion process and to find a method which will solve the problem both accurately and efficiently. To this end the method of moving finite elements (MF) is used and various modifications to the original MF method [8] are examined, including the use of penalty functions, weight functions and graph massage[2]. A method for dealing with the second numerical difficulty mentioned above, by means of a transformation, is discussed and the results from the FORTRAN 90 code are presented in Chapter 5. Finally, conclusions are drawn about the various strategies investigated.

[1]Before the dopant is implanted a very small concentration of dopant of opposite sign is distributed uniformly throughout the silicon. The junction is then defined as the place where the concentration of the implanted dopant drops below this background level.

[2]This is a new approach for dealing with the problems of node migration to and from certain regions of the graph and is dealt with in some detail in Chapter .

Ch pter 2

he Semiconductor roblem

2.1 The Diffusion Model

2.1.1 General Case

It is necessary to start by formulating a model of the non-linear diffusion process that takes place within the crystalline silicon as the dopant diffuses. The details of the diffusion mechanisms involved are omitted here, but they can be found in [3] and [6]. An equation for the conservation of the dopant concentration c is obtained, and can be written in the form

$$\frac{\partial c}{\partial t} = \nabla.(\quad (c)\nabla c) \tag{2.1}$$

where

$$\left.\begin{aligned} (c) &= \quad {}_i[\tfrac{1+\beta\frac{n_e}{n_i}}{1+\beta}] \\ n_e &= \tfrac{1}{2}[\, c + \sqrt{c^2 + 4n_i^2}\,] \end{aligned}\right\} \tag{2.2}$$

β, ${}_i$ and n_i are constants and the implantation of the dopant is modelled by taking the initial profile to be a Gaussian hump which has a standard deviation

that is small in comparison to the distance over which the the dopant diffuses.

quations (2.1) and (2.2) can be simplified further to obtain the *model* problem

$$\left.\begin{array}{rcl} \frac{\partial c}{\partial t} & = & \nabla.(\quad (c)\nabla c) \\ (c) & = & c+\epsilon \end{array}\right\} \tag{2.3}$$

where $\epsilon \quad 1$ (typically $O(10^{-2})$). This new equation has the essential features of equations (2.1) and (2.2) and taking a Gaussian initial profile leads to similar behaviour to that of the full problem.

2.1.2 The 1-D Case

In 1-D, equation (2.3) becomes

$$\frac{\partial c}{\partial t} = \frac{\partial}{\partial x}\left((c+\epsilon)\frac{\partial c}{\partial x}\right) \tag{2.4}$$

and carrying out the differentiation on the RHS gives

$$\frac{\partial c}{\partial t} = \left(\frac{\partial c}{\partial x}\right)^2 + (c+\epsilon)\frac{\partial^2 c}{\partial x^2} \tag{2.5}$$

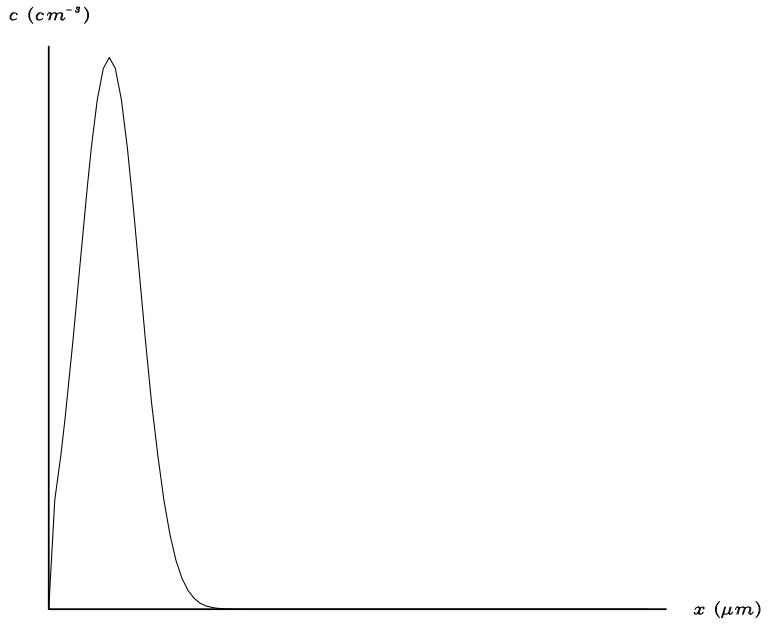
The boundary conditions are taken to be

$$\left.\begin{array}{rcl} \frac{\partial c}{\partial x} & = & 0 \; at \; x \; = \; -a \\ c & \rightarrow & 0 \; as \; \mid x \mid \rightarrow \infty \end{array}\right\} \tag{2.6}$$

and the initial condition is a Gaussian hump, perpendicular to the surface of the silicon, given by

$$c \; = \; \lambda e^{\frac{-(x-\omega)^2}{2\sigma^2}} \; at \; t \; = \; 0 \tag{2.7}$$

where



Fig

Detailed information needs to be provided in the tail region in order to accurately

Using equation (2.9) with the model problem (2.3), where $(c) = c + \epsilon$, gives the new variable

$$\phi = c + \epsilon \, log_e(c) \tag{2.10}$$

For small c, $\phi \sim \epsilon \, log_c(c)$, so c has been stretched logarithmically in this region (and only in this region), which allows greater resolution in the tail where the solution is no longer exponential but quadratic (since the initial function is quadratic for small c under this transformation). For large c, $\phi \sim c$, so the behaviour of the solution is basically unaltered in this region and this is needed to model the early dominance of the diffusion mechanism.

2.2.3 Application to the Semiconductor Problem

Substituting equation (2.8) into equation (2.1) gives

$$\frac{\partial c}{\partial t} = \nabla.(c \nabla \phi) \tag{2.11}$$

and differentiating (2.10) with respect to t gives

$$\frac{\partial \phi}{\partial t} = \frac{(c)}{c} \frac{\partial c}{\partial t} \tag{2.12}$$

Substituting equation (2.11) into equation (2.12) gives

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= \frac{(c)}{c} \nabla.(c \nabla \phi) \\ &= \frac{(c)}{c} \left[\nabla c \nabla \phi + c \nabla^2 \phi \right] \\ &= \frac{(c)}{c} \nabla c \nabla \phi + \quad (c) \nabla^2 \phi \end{aligned} \tag{2.13}$$

Rearranging equation (2.8) and substituting into equation (2.13) gives

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= \nabla \phi \nabla \phi + \quad (c) \nabla^2 \phi \\ &= (\nabla \phi)^2 + \quad (c) \nabla^2 \phi \end{aligned} \tag{2.14}$$

In 1-D equation (2.14) becomes

$$\frac{\partial \phi}{\partial t} = \quad \frac{\partial \phi}{\partial t} \quad ^{2} + \quad (\ ) \frac{^{2}}{^{2}} \tag{2 15}$$

and this does not look very different to equation (2.5). However, the advantage of equation (2.15) is the scale of when compared to that of , and the severity of the numerical difficulties is greatly reduced.

Although appears in equation (2.15) which is a PD for , it is a simple task to calculate from using (2.10). The FORTRAN 90 code uses Newton iteration to carry out this inversion which is computationally inexpensive: typically only three or four iterations are required, although more iterations are required at certain values[1] of .

It is worth noting that for the full problem, equations (2.1), (2.2) and (2.9) imply that a much more complicated relationship than (2.10) exists between the variables and , but the model diffusion coefficient of (2.3), namely ( ) = + , gives the qualitative scaling that is required. Hence, it is sufficient for this study to use (2.10) and avoid the complications of using ( ) as in (2.2).

[1]These *critical* values are around the point where the initial approximation changes from $c = \phi$ to $c = e^{\frac{\phi}{\epsilon}}$, but even at these values the Newton solver takes only six or seven iterations to converge.

Ch pter 3

he Finite Element Method

3.1 Reasons for Using Finite Elememts

In Chapter 1 it was stated that one of the aims of the dissertation was to find a method that is able to accurately track and model the position of the moving front. On a fixed grid it would be necessary to have a very fine grid to provide a good resolution of the moving front, but away from the front the solution is smooth and the fine grid is not necessary. To avoid the computational inefficiency of using a large number of grid-points, the finite element method (F M) is used since it is able to provide the irregular grid that is needed for this problem at any fixed time. Furthermore, if the grid is allowed to move then it is hoped that the 'correct' concentration of nodes will be provided at the required positions.

3.2 The Fixed Finite Element Method

Consider the general PD

$$\frac{\partial c}{\partial t} = \mathcal{L}\, c \tag{3.1}$$

where $\mathcal{L}$ is a spatial differential operator which involves, at most, second order derivatives. A variational principle can be obtained for equation (3.1) by considering the weighted L_2 norm of the PD which is

$$R = \int_a^b \left(\frac{\partial c}{\partial t} - \mathcal{L}\, c \right)^2 w \, dc \tag{3.2}$$

where w is a weight function to be chosen later. If R is minimised with respect to $\frac{\partial c}{\partial t}$ (by setting $\frac{\partial R}{\partial c_t} = 0$) then equation (3.1) is recovered. This procedure can be extended to the case where c is replaced by an approximation and an intuitive interpretation is that R measures the residual, ie the degree to which the original equation fails to be satisfied.

The standard Galerkin method of finite elements is derived by approximating c by

$$c\,(x\,,t\,) = \sum_{j=1}^{n} c_j(t)\,\alpha_j(x) \tag{3.3}$$

where

$c_j(t)$ are the time-dependent amplitudes,

$\alpha_j(x)$ are the time-independent basis functions.

and the time derivative of c, from equation (3.3), is given by

$$\frac{\partial c}{\partial t} = \sum_{j=1}^{n} \dot{c}_j(t)\,\alpha_j(x) \tag{3.4}$$

Discretised equations for the nodal amplitudes can be found by minimising R with respect to the time derivatives of the amplitudes, ie by setting $\frac{\partial R}{\partial \dot{c}_j} = 0$. This generates a set of equations for the c_j which give the *'best fit'* of $\frac{\partial c}{\partial t}$ to $\mathcal{L}\, c$ and these are given explicitly by

$$\langle \alpha_i\,,\ \frac{\partial c}{\partial t} - \mathcal{L}\, c \rangle = 0\,, \quad i = \{1, \ldots, n\}$$

$$\alpha_i\,,\ \sum_{j=1}^{n}\dot{c}_j\,\alpha_j \qquad c\ =0\,,\quad i=\ 1,\ldots,n$$

$$\alpha_i\,,\ \sum_{j=1}^{n}\dot{c}_j\,\alpha_j\ =\ \alpha_i\,,\quad c\ ,\quad i=\ 1,\ldots,n$$

$$\sum_{j=1}^{n}\alpha_i\,,\ \alpha_j\ \dot{c}_j=\ \alpha_i\,,\quad c\ ,\quad i=\ 1,\ldots,n \tag{3.5}$$

where

$$=\ \int_a^b$$

quation (3.5) represents a system of OD s in unknowns (the nodal amplitudes), so the original PD problem has been reduced to an OD system and the integration of these OD s gives a continuously evolving function which approximates the solution of (3.1).

Conventionally, finite element basis functions are chosen so that they vanish outside of a fixed interval, which results in the matrix given by the inner products $_i\ \ _j$ in equation (3.5) being banded or sparse, since distant basis functions do not overlap. This simplifies the task of evaluating the inner products (and inverting the matrix if necessary). The most common finite elements that are used are finite elements and it is these that are used in the following work.

A grid of points, $_j$ is defined such that $=\ _0\ \ \ldots\ \ _j\ \ \ldots\ \ _n=\ .$ Then each basis function, $_j$, is defined to be 1 at $_j$, linear in the interval $[\ _{j-1}\ \ _{j+1}]$ and 0 outside the interval $[\ _{j-1}\ \ _{j+1}]$. More explicitly, the basis functions are given by

$$\alpha_j = \begin{cases} \frac{x - s_{j-1}}{s_j - s_{j-1}} & x \in [s_{j-1}\,,\, s_j] \\ \frac{s_{j+1} - x}{s_{j+1} - s_j} & x \in [s_j\,,\, s_{j+1}] \\ 0 & otherwise \end{cases} \tag{3.6}$$

and a typical basis function α_j can be seen in Figure 3.1. With this choice of

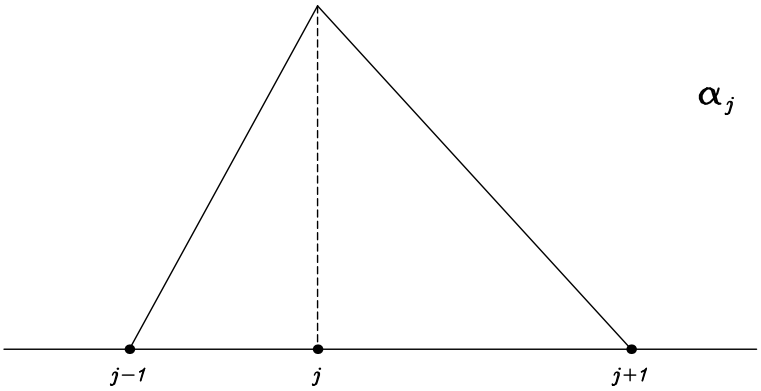


Figure 3.1

basis functions the matrix resulting from the inner products of equation (3.5) is tri-diagonal.

3.4 Problems of the Fixed FEM

While the F M discussed above can provide a non-uniform grid and thus reduce the number of nodes by concentrating nodes in regions where the solution gradient varies rapidly and placing fewer nodes in regions where the solution is smooth, it cannot adapt itself to a problem where these regions move in time. In order to track these moving regions it is necessary to allow the grid to adapt as the solution evolves. From the point of view of the F M it is desired to find a way in which the basis functions can adapt themselves to the evolving solution. It is

for this reason that the method of moving finite elements (MF) was introduced and it is this method that is examined here in some detail.

3.5 The Moving Finite Element Method

The MF method is an extension of the fixed F M where the nodes are allowed to move over time, and this is an example of a dynamic rezoning strategy. The grid positions s_j become functions of time and c is now approximated by

$$c(x\,,\,t) = \sum_{j=1}^{n} c_j(t)\,\alpha_j(x\,,\,s_j(t)) \tag{3.7}$$

The time derivative of c is much more complicated than for the fixed F M owing to the dependence of α_j on the nodes s_{j-1}, s_j and s_{j+1}, which are all free to move in time.

3.5.1 Derivation of $\frac{\partial c}{\partial t}$ for the MFE

Differentiating equation (3.7) with respect to t gives

$$\frac{\partial c}{\partial t} = \sum_{j=1}^{n} \left[\dot{c}_j(t)\,\alpha_j(x\,,\,s_j(t)) + c_j(t)\,\frac{\partial}{\partial t}\alpha_j(x\,,\,s_j(t)) \right] \tag{3.8}$$

Now,

$$\begin{aligned} \sum_{j=1}^{n} c_j(t)\,\frac{\partial}{\partial t}\alpha_j(x\,,\,s_j(t)) &= \sum_{j=1}^{n} c_j \frac{\partial \alpha_j}{\partial \mathbf{s}} \frac{d\mathbf{s}}{dt} \\ &= \sum_{j=1}^{n} c_j \sum_{i=1}^{n} \frac{\partial \alpha_j}{\partial s_i}\dot{s}_i \end{aligned} \tag{3.9}$$

Consider a general basis function α_j for a piecewise linear approximation. Clearly, $\frac{\partial \alpha_j}{\partial s_i}$ is only non-zero for $s_i \in \{s_{j-1},\, s_j,\, s_{j+1}\}$.

$$\frac{\partial \alpha_j}{\partial s_{j-1}} \quad j \quad 1 \qquad \frac{(s_j \quad x)}{(s_j \quad s_{j-1})^2} \quad j \quad 1$$

$$\frac{\partial \alpha_j}{\partial s_j} \quad j \qquad \frac{(x \quad s_{j-1})}{(s_j \quad s_{j-1})^2} \qquad \frac{(s_{j+1} \quad x)}{(s_{j+1} \quad s_j)^2} \qquad j$$

$$\frac{\partial \alpha_j}{\partial s_{j+1}} \quad j{+}1 \qquad \frac{(x \quad s_j)}{(s_{j+1} \quad s_j)^2} \quad j{+}1$$

$n \qquad n \qquad \frac{j}{i} \quad i \qquad iRRRRiRRjRRRjRiRRieieieiei$

$j \qquad$

$j{=}1 \qquad i{=}1 \qquad j{=}1$

$$\frac{+1}{+1} \quad {}^{2} \quad +1 \qquad \frac{+1}{}$$

$$\frac{1}{1} \quad {}^{2} \qquad \frac{+1}{+1} \quad {}^{2} \qquad \frac{}{}$$

$$\frac{}{1} \quad {}^{2} \quad 1 \qquad \frac{1}{}$$

$$\frac{+1}{+1} \quad \frac{+1}{+1} \qquad \frac{1}{1} \quad \frac{1}{1}$$

1

$+1 \qquad\qquad +1$

$$\frac{1}{1}$$

$$\frac{}{} \quad \frac{}{}$$

$=1 \qquad\qquad =1$

$$\frac{}{}$$

$=1$

The β_j's will be discontinuous (since the m_j's can be regarded as a discontinuous function) and a typical basis function β_j can be seen in Figure 3.2.

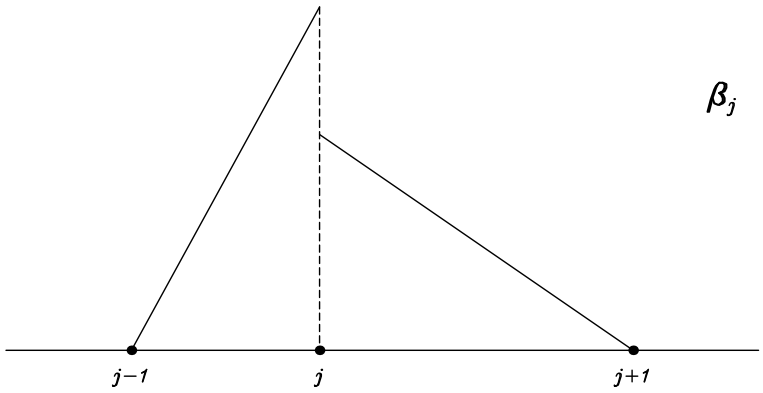


Figure 3.2

3.5.2 Minimisation of R

If the residual, R, in equation (3.2) is now minimised with respect to $\dot{c}_j$ and $\dot{s}_j$, by setting $\frac{\partial R}{\partial \dot{c}_j} = 0 = \frac{\partial R}{\partial \dot{s}_j}$, then this generates movements which give the *'best fit'*, in the same sense as before, over the enlarged space of nodal amplitudes and nodal velocities, and therefore, it is claimed, a better fit, since the grid moves in such a way as to minimise the residual and hopefully this will optimise the solution. The validity of this claim is somewhat questionable though, as the node movement may have more to do with the 'crudeness' of using linear basis functions than anything else, and details of this can be found in [1].

The result of the minimisation of R in this way is referred to as a method of lines (MOL) and the equations are given more explicitly by

$$\left.\begin{array}{rcl} \langle \alpha_i \,,\, \frac{\partial c}{\partial t} - \mathcal{L}\, c \rangle & = & 0 \\ \langle \beta_i \,,\, \frac{\partial c}{\partial t} - \mathcal{L}\, c \rangle & = & 0 \end{array}\right\} \quad i = \{1, \ldots, n\}$$

$$
\begin{aligned}
i \quad & \sum_{j=1}^{n} [\ j \dot{}_j + \ j \dot{}_j] \quad = \ 0 \\
i \quad & \sum_{j=1}^{n} [\ j \dot{}_j + \ j \dot{}_j] \quad = \ 0
\end{aligned}
\qquad = \ 1 \ \ldots
$$

$$
\begin{aligned}
\sum_{j=1}^{n} \ i \quad j \dot{}_j + \sum_{j=1}^{n} \ i \quad j \dot{}_j \ &= \quad i \\
\sum_{j=1}^{n} \ i \quad j \dot{}_j + \sum_{j=1}^{n} \ i \quad j \dot{}_j \ &= \quad i
\end{aligned}
\qquad = \ 1 \ \ldots \qquad (3\ 17)
$$

quations (3.17) represent a system of 2 OD s in 2 unknowns and the matrix associated with the inner products of (3.17) is 2 x 2 block tri-diagonal.

If is the matrix associated with the inner products of (3.17) then each 2 x 2 block, $_{ij}$ (= 1 ...), of is given by

$$
_{ij} = \begin{matrix} i & j & i & j \\ i & j & i & j \end{matrix} \qquad = \ 1 \ \ldots \qquad (3\ 18)
$$

$$
i \quad j \qquad i \quad j
$$

---

i j j j $\frac{1}{3}$ j $j+1$

i j j j $\frac{1}{3}$ j $j+1$

i j j j $\frac{1}{3}$ j j $j+1$ $j+1$

min max

max

min

–1

1

2

[1]A tolerance of 10^{-8} was used to test the difference between succesive iterations.
[2]The number of iterations depends on the value of the parameter δ.

3.8 Parallelism

Unfortunately, the MF method is not without its problems. One such problem is that of *parallelism* and this occurs when the equations in the system (3.23) become singular (or nearly singular) at any point. This is a problem because the *'best fit'* is now ambiguous and this creates difficulties for the OD solver. A singularity occurs when the solutions on adjacent cells lie on the same line, but it is possible to move the offending node without affecting the MF approximation. Thus, the singularity can be removed by specifying any consistent value for the motion of the node.

3.8.1 Penalty Functions

An alternative approach to dealing with parallelism is to use *penalty functions* which have the effect of adding a small positive definite term

$$R_{+ve} = \epsilon \sum_{j=1}^{n} \frac{(\dot{s}_j - \dot{s}_{j-1})^2}{(s_j - s_{j-1})} \tag{3.24}$$

to the residual R in (3.2), where R_{+ve} only depends on the nodal positions and their time derivatives [4]. The new residual becomes

$$R_p = R + R_{+ve} = \int_a^b \left(\frac{\partial c}{\partial t} - \mathcal{L}\, c \right)^2 w \, dx + \epsilon \sum_{j=1}^{n} \frac{(\dot{s}_j - \dot{s}_{j-1})^2}{(s_j - s_{j-1})} \tag{3.25}$$

and applying the condition $\frac{\partial R_p}{\partial \dot{s}_j} = 0 = \frac{\partial R_p}{\partial \dot{c}_j}$ gives rise to some new equations from the $\frac{\partial R_p}{\partial \dot{s}_j} = 0$ condition, but not from the $\frac{\partial R_p}{\partial \dot{c}_j} = 0$ condition, since R_{+ve} does not depend explicitly on $\dot{c}_j$.

3.8.2 Changes to the Matrix A

Consider $\frac{\partial R_{+ve}}{\partial \dot{s}_j} = 0$. Then for any j,

$$\frac{\partial}{\partial \dot{s}_j} \epsilon \sum_{j=1}^{n} \frac{(\dot{s}_j - \dot{s}_{j-1})^2}{(s_j - s_{j-1})} = 0$$

$$\Rightarrow \epsilon \left[\frac{(\dot{s}_j - \dot{s}_{j-1})}{(s_j - s_{j-1})} - \frac{(\dot{s}_{j+1} - \dot{s}_j)}{(s_{j+1} - s_j)} \right] = 0$$

$$\Rightarrow \epsilon \left[-\frac{1}{(s_j - s_{j-1})} \dot{s}_{j-1} + \left(\frac{1}{(s_j - s_{j-1})} + \frac{1}{(s_{j+1} - s_j)} \right) \dot{s}_j - \frac{1}{(s_{j+1} - s_j)} \dot{s}_{j+1} \right] = 0$$

So, minimising the new residual (3.25) with respect to $\dot{s}_j$ and $\dot{c}_j$ has the effect of adding small positive definite terms to the matrix **A** in (3.17) and the changes to this matrix are given by:

Changes on diagonal block

$$\langle \beta_i \, , \, \beta_j \rangle \rightarrow \langle \beta_j \, , \, \beta_j \rangle + \epsilon \left[\frac{1}{(s_j - s_{j-1})} + \frac{1}{(s_{j+1} - s_j)} \right] , \quad i = \{1, \ldots, n\}$$

Changes on super-diagonal block

$$\langle \beta_i \, , \, \beta_j \rangle \rightarrow \langle \beta_{j+1} \, , \, \beta_j \rangle - \frac{\epsilon}{(s_{j+1} - s_j)} , \quad i = \{1, \ldots, n\}$$

The effect of adding this *internodal viscosity* is to restrain the node motion in regions where the usual restoring force on a node becomes small and the optimum choice of the constant ϵ is given by Miller to be ten times the square of the desired truncation error. This should be large enough to avoid numerical difficulties, but small enough so that grid motion will not be affected in regions where the original matrix is not nearly singular.

The FORTRAN 90 code makes use of both of these procedures to deal with the problem of parallelism, but as parallelism wasn't a difficulty that was encountered with the semiconductor problem it is hard to assess the usefulness of the penalty functions.

While the MF method is expected to cause the grid to concentrate in regions of steep gradient, there is also a tendency for most (or all) of the nodes to concentrate in the region of the moving front, leaving the rest of the region with very few (or no) nodes. ven worse is the case where nodes overtake each other causing a tangling of the grid, but if a "sensible" time-step is taken then this shouldn't

$\frac{2}{j}$ $-\frac{1}{2}$

j $\frac{\partial c}{\partial x}$

weight function does not have the ability to de-emphasize regions of the graph as required and a non-uniform weight function destroys the exact conservation property. Therefore, a decision has to be made as to which property is 'least' desired, and if a non-uniform weight function is used, then it is hoped that the additional accuracy arising from an improved nodal distribution will offset the loss of the exact conservation property.

3.11 Recovery

When solving equations (3.17) it is necessary to evaluate the inner product

$$\langle \beta_i \, , \, \mathcal{L}\, c \rangle \tag{3.27}$$

However, since $\mathcal{L}$ contains second derivatives which don't exist and the β_i are discontinuous[3], this inner product does not exist and it needs to replaced by

$$\langle \beta_i \, , \, \mathcal{L}\, (\mathcal{S}\, c) \rangle \tag{3.28}$$

where $\mathcal{S}$ is a smoothing operator defined explicitly as $\mathcal{S}\, c(x) = v(x)$ [5] by constructing a *recovered* function $v(x)$ from the piecewise linear approximation to c, or its gradient $\frac{\partial c}{\partial x}$, which has sufficient continuity to allow the evaluation of (3.28). Suitable functions $v(x)$ may be constructed by fitting local polynomials of sufficiently high order to c or $\frac{\partial c}{\partial x}$.

It can be shown that taking a Hermite cubic interpolation between nodes i and $i-1$ is equivalent to δ-mollification (ie, arbitrary smoothing of c) and since $v(x)$ can be defined in a variety of other ways as well, the recovery procedure

[3] Usually, integration by parts copes with the problems of second derivatives in finite element approximation, but the discontinuity in the β_i prevents this.

offers greater flexibility than the other methods with no loss of computational efficiency[4]. The use of recovery can also improve accuracy and allow larger time-steps and it is for these reasons that the FORTRAN 90 code uses the recovery procedure.

A quadratic function $\frac{\partial v}{\partial x}$ is fitted to the MF gradient $\frac{\partial c}{\partial x}$ to exploit ideas of

$$\frac{\partial v}{\partial x} \quad i-1 \qquad\qquad i-1 \qquad i$$

$$\frac{\partial v}{\partial x} \quad \frac{1}{2} \quad i-1 \qquad i \qquad\qquad i$$

$$\frac{\partial v}{\partial x} \quad i \qquad\qquad i \qquad\qquad i+1$$

$$\frac{\Delta s_i}{(\Delta s_i + \Delta s_{i-1})}$$

$$\frac{\Delta s_i}{(\Delta s_i + \Delta s_{i+1})}$$

— —

— ———

— —

— — — —

2

— — —

2

[4]This is only true in the 1-D case. In 2-D, the recovery procedure becomes a lot more complicated and may not be the best way of evaluating the inner products.

and it should be noted that in equation (3.29) all of the terms which involve the smoothing operator and are differentiated with respect to x need to be recovered, namely $\frac{\partial}{\partial x}(\quad(v))$ and $\frac{\partial^2 v}{\partial x^2}$. While the recovery of $\frac{\partial}{\partial x}(\quad(v))$ is not *strictly* necessary, it does help with the nodal velocities and should be used in practice as better numerical results can be obtained.

While the instantaneous movement of the nodes using the gradient weighted MF (GWMF) method are optimal with respect to the minimisation of the residual R, it is sometimes noted that node depletion and excessively long line segments can gradually occur in certain regions of the solution graph over 'long' periods of time. Conversely, moving fronts can push too many nodes into small regions of the solution graph over 'long' periods of time. To remedy this problem a technique has been developed by Andrew Kuprat which assesses the GWMF solution graph at regular intervals and adds or deletes nodes where appropriate. This technique is called *raph Massage* [7]. Owing to the continuous movement of the nodes it is expected that the solution graph will be massaged fairly infrequently and then, that only a small number of nodes will be added or deleted at any time.

The graph massage algorithm is a FORTRAN 90 subroutine which creates nodes according to two criteria (and) and annihilates nodes using one

4.2.2 Perturbational Stability (PS)

Assume that graph massage has been performed on a graph and that the resultant graph is G with an associated solution vector $\mathbf{c}$. Now suppose that G is perturbed to a new graph G' with the corresponding solution vector $\mathbf{c}' = \mathbf{c} + \boldsymbol{\epsilon}$. Then graph massage should refuse to alter G' in any way provided that $\boldsymbol{\epsilon}$ is sufficiently small and this is called *Perturbational Stability*.

This stability is important when $\mathbf{c}$ represents the GWMF approximation to a time-dependent PD , because the graph massage routine will be called every few time-steps and small perturbations in $\mathbf{c}$ will occur between each call owing to the evolutionary nature of the PD . Therefore, to avoid computational inefficiencies arising from changes made to the graph, it is desired that graph massage will only make changes to G occasionally.

In practice, perturbational stability is achieved by working with two sets of tolerances. After the user has input various tolerances, graph massage increases or decreases each tolerance by a user-specified stability factor so that graph massage is discouraged. If, however, creation or annihilation is required with these 'harder' *trigger* tolerances, then all of the tolerances are relaxed to their original values and the graph massage modules are called. The output solution vector $\mathbf{c}$ now satisfies the original tolerances as well as the *trigger* tolerances which are harder to violate, so perturbational stability is achieved.

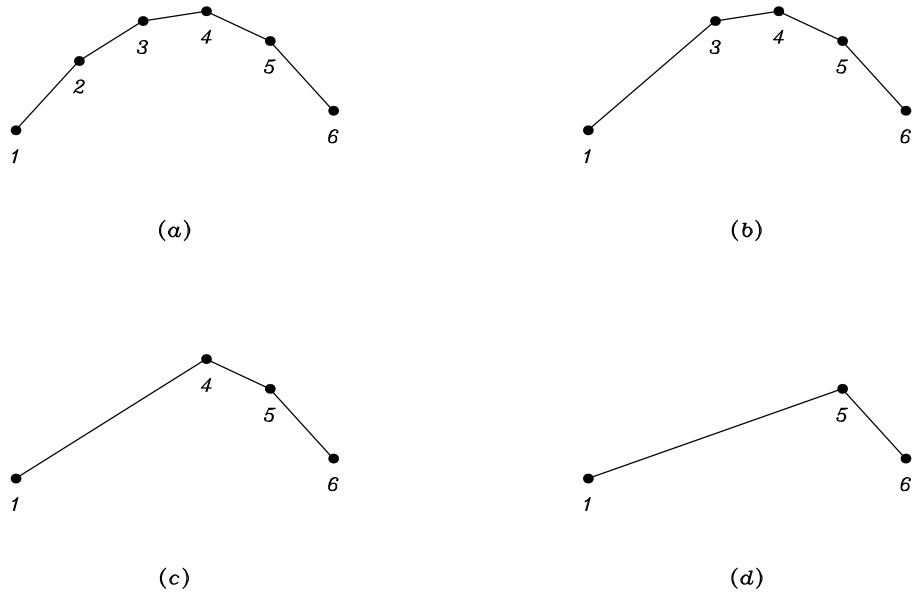


Figure 4.1 Unaccept ı

2) The angles between the normals to adjacent faces of the graph

chang by a bounded amount when mapping to the new graph ′.

This means that the points on are only allowed to move a small distance and that the shape of the graph is only allowed to be distorted by a small amount.

i i

i i

i $i+1$ i i

If a given segment, $_i$, is longer than the user-specified maximum segment length, _ _ , then this module attempts to insert a node. This node will be inserted so as to bisect $_i$, as in Figure 4.2.

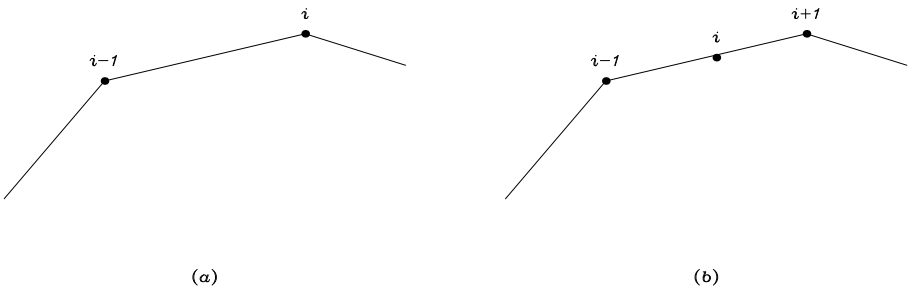


Figure 4.2 Insertion of node by Creation on Length.

The insertion of a node is not guaranteed though, and the failure to add a new node may be owing to one of the following reasons:

i $i+1$

_ _ _ _

exceeds _ _ . This list is then ordered[1] smallest to largest, and segments are taken from the end of the list. If a new node is added then the new segment lengths are checked to see if they exceed _ _ , and if they do then they are added to the list.

i

i

i

_ _

[1]Every time that nodes are added to the list it needs to be sorted. This should not cause any problems though, as it is expected that relatively few nodes will be created or annihilated at any given time.

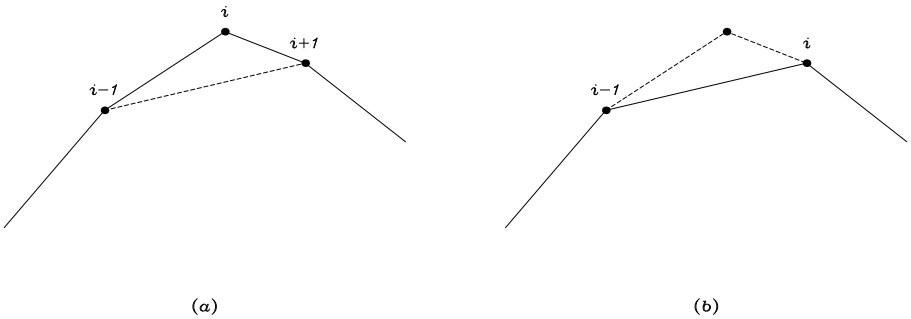


Figure 4.

While the annihilation list contains the nodes, and the damage that the nodes would do to the graph if removed, it does not in any way measure the damage that the graph would suffer as a result of removing a node, taking into account any previous annihilations around that node. This damage

– – i–1 – – i–1 – – i i

– – i–1 – – i–1 i – – i i

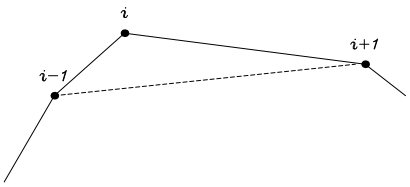
i

i – –

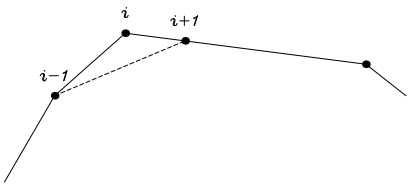
i

– –

i



(a)



(b)

F'

One useful feature of FORTRAN 90 that was utilised by the code was that of derived data types, which made it easy to keep track of the nodes that were associated with particular segment lengths, break values and angles in the creation/annihilation routines. In 2-D, where the data structure is more complicated, the use of derived data types would be particularly useful. The use of FORTRAN 90 also made the general coding process a lot easier.

max

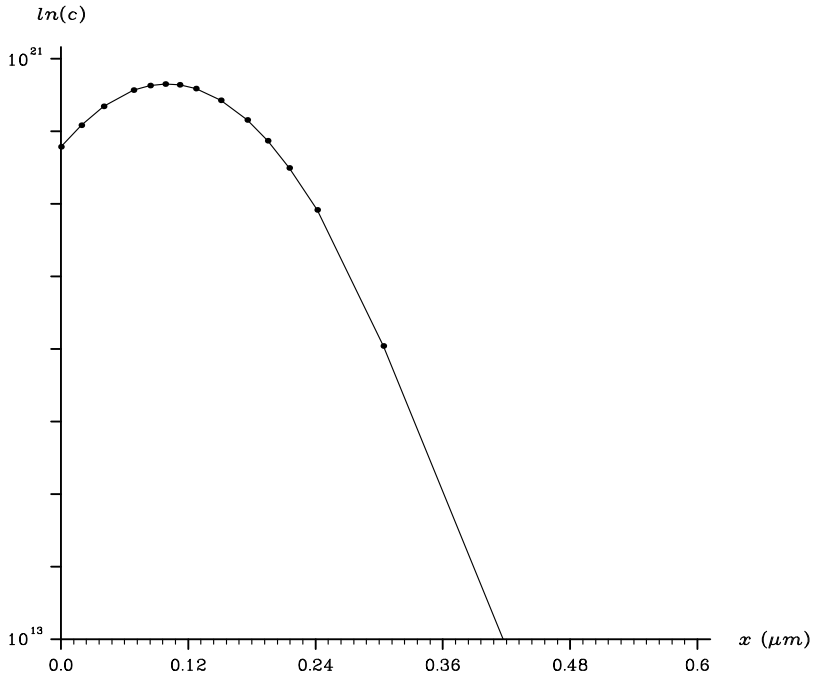
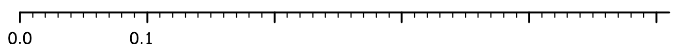


Figure 5.


0.0
0.1


0.

Ch pter 6

Conclusions

In this dissertation the method of moving finite elements has been examined, and then applied, to an existing problem in semiconductor process modelling. A program had already been written for the problem several years ago, by P.K. Sweby, but it had been observed that the program wasn't very *robust* in the sense that for more "difficult" cases (ie a different choice of parameters) the program failed to resolve the problem accurately.

The original program was re-written in FORTRAN 90 and then various modifications were made to it in an attempt to improve the robustness of the method. The original problem was retained in order to assess any improvements that might result.

The first modification was the use of weight functions which give rise to the GWMF method. It was hoped that the gradient weighting would improve the nodal distribution by moving nodes away from the steep front and into the tail region where the resolution is of great importance. However, when comparing Figures 5.1 and 5.2, which are the normal and gradient weighted

outputs respectively, it is not obvious that there is any difference between the two solutions.

Comparison of the nodal positons over time, Figures 5.3 and 5.4, also suggests that GWMF has not performed any better than normal MF as the nodal movement is almost identical. The fact there was no nodal movement away

[1]Since the solution did not become parallel when run using the normal MFE method, it was not expected that penalty functions would greatly improve the solution.

and although several of these can be related to each other, at least *four* of the parameters have to be chosen by trial and error. It took several hours of experimention before any parameters were found to work for the semiconductor problem, and even then the results were rather disappointing.

Two plots using graph massage can be seen in Figures 5.7 and 5.8. The first appears to give quite a good resolution of the shock, but there is something strange happening around the tail region. The second appears to give a poorer approximation to the shock, but the 'kink' in the tail region is less pronounced. No amount of fiddling with parameters managed to produce a solution that looked much different from the two shown here.

There is also a "problem" associated with graph massage and this is the fact that the height of the solution should be of the same order as the length of the solution interval. If this is not the case, and the solution height is a lot larger than the the interval length, then the 'dumb' creation on length module will require either a large value for *Max_Segment_Length*, in which case nodes will tend to be placed on any peak in the solution and nowhere else, or the number of nodes will have to be large in order to make *Max_Segment_Length* small enough to resolve details away from any peak in the solution. The result is that the it may be necessary to scale the problem in order to use *raph Massage* and again this is not a good thing for robustness.

Although the results obtained have not been very promising, the use of graph massage in 2-D would seem to offer the possibility of solving ('fixing') problems, which until now, may have had little chance of being solved. If more time had been available, an investigation of graph massage in 2-D would have been undertaken,

and it is in 2-D that Graph Massage appears to have most applicability.

[1] M.J. Baines, N.R.C. Birkett P.K. Sweby (1990),

, , Vol. 3, pp. 79 - 90.

[2] P. Concus, G.H. Golub, D.P. O'Leary (1975),

, in

(ds. J.R. Bunch, D.J. Rose) Academic Press, New York, pp. 309 - 332.

[3] R.B. Fair (1981), , in

(d. F.F.Y. Wang) North Holland, New York, pp. 315 -442.

[4] A.H. Glasser (1988),

, , Vol. 85 N°1, pp. 159 - .

o

[7] A.P. Kuprat (1992), *'Creation and Annihilation of Nodes for the Moving Finite Element Method'*, **PhD Thesis**, University of California.

[8] K. Miller (1981), *'Moving Finite Elements I'*, **SIAM J. Numer. Anal.**, 18, pp. 1019 - 1032.

[9] R.O. Moody (1988), *'The Numerical Solution of Moving Boundary Problems Using MFE Methods'*, **PhD Thesis**, University of Reading.

[10] C.P. Please P.K. Sweby (1986), *'A Transformation to Assist Numerical Solution of Diffusion Equations'*, **Numerical Analysis Report 5/86**, University of Reading.