Adaptive remeshing and $h$-$p$ domain decomposition

Ernst Rank

Numerische Methoden und Informationsverarbeitung, FB Bauwesen, University of Dortmund, Germany

Received 14 October 1991

The $h$-$p$ version of the finite element method combines local mesh refinement with an increase of the shape functions' polynomial degree. This paper presents a method which uses an $h$-$p$-approximation for a domain decomposition with fully overlapping domains, allowing $C^0$-continuous approximation for geometrically incompatible mesh refinement to be obtained. The method is applied to a reaction-diffusion problem resulting from semiconductor process simulation. In numerical examples, it is demonstrated that this approach is well suited for problems with sharp interior layers or shock-like behaviour.

1. Introduction

Since the first publications on the $h$-$p$ version of the finite element method [1, 2], a lot of progress has been achieved in research on basic theoretic principles (e.g. [3, 4]), new applications (e.g. [5, 6]) and implementation aspects (e.g. [7-9]). The high attractiveness of the $h$-$p$ version is due to its robustness, its extreme accuracy and to its natural connections to adaptive strategies. The smart combination of mesh refinement ($h$-version) and increase of the shape functions' polynomial degree ($p$-version) yield, for linear elliptic boundary value problems, an exponential rate of convergence in the energy norm even if the exact solution is singular. This has been demonstrated theoretically [3, 4] and in many numerical examples [5-9].

There are two basic observations making the $h$-$p$ version work. The first is the $p$-version's asymptotic exponential rate of convergence in the energy norm for an analytic exact solution. Secondly it can be shown that the $p$-version on a fixed, geometrically refined mesh converges exponentially in some pre-asymptotic range. If a successive mesh refinement at singular points is combined with a proper global increase of the polynomial order, an overall exponential rate of convergence is achieved. It has also been shown [4, 8] that an optimal geometric mesh refinement is obtained for a progression factor of 0.85 yielding a strongly decreasing size of elements near singular points.

Correspondence to: Ernst Rank, Universität Dortmund, Fachbereich Bauwesen, August-Schmidt-Strasse, 4600 Dortmund Eichlinghofen, Postfach 50 05 00, Germany.

0045-7825/92/$05.00 © 1992 Elsevier Science Publishers B.V. All rights reserved
successful implementation of an $h$-$p$ version, some additional problems have to be solved. It is unclear how a proper combination of $h$-refinement and $p$-increase can be chosen and it is geometrically more involved to achieve a strong mesh grading towards singular curves.

In this paper, the second problem, in particular, is addressed by a combination of the $h$-$p$ version with a domain decomposition method. The paper is outlined as follows: in Section 2 a reaction-diffusion problem in semiconductor process simulation with a very sharp reaction front is presented. Section 3 describes the new approach, extending methods developed by Ewing, Bramble and others (e.g. [10–12]) for low order finite element models and regular grids. In Section 4, implementation aspects are discussed and in Section 5 numerical examples are presented.

2. A reaction-diffusion problem in semiconductor process simulation

Local oxidation of silicon is a process step in microelectronic chip fabrication used to isolate adjacent electronic devices (transistors, capacitors, etc.) against each other.

Figure 1 shows a simplified three-dimensional view of a typical structure in highly integrated electronic circuits. Ranges marked as ‘source’ and ‘drain’ are electrically active zones of adjacent transistors; the ‘field oxide’ between the two source regions is set up to isolate the two ranges against each other. The width of this oxide, which grows during local oxidation of silicon, is typically 1000 nm. The corresponding process step is explained in the two-dimensional cross section through the structure as shown in Figs. 2 and 3.

After a thin layer of planar oxide has been deposited onto the silicon, the structure is masked partly by silicon nitride (Fig. 2). Then the wafer is oxidized in an atmosphere of oxygen (dry oxidation) or steam (wet oxidation) at temperatures between 900 °C and 1200 °C. During this process step, the oxidant penetrates the unmasked surface of SiO$_2$ and diffuses through the oxide to the interface with silicon. There, a chemical reaction of oxidant and silicon to oxide has two consequences: firstly, silicon is consumed, so the interface moves during oxidation; secondly, the oxide produced has more than twice the volume of the original silicon – the oxide layer is lifted up (Fig. 3).

A direct mathematical modelling of the local oxidation of silicon resulting in a moving boundary–moving interface problem was formulated first in [13] and is discussed for example in [14] or [15]. This basic model is used in process simulation programs (e.g. [15, 16]). Essentially, only the silicon dioxide layer is discretized and its boundary is geometrically updated in a cycle of diffusion and structural deformation steps.

Fig. 1. Three-dimensional view of adjacent electronic devices.

Fig. 2. Initial domain for oxidation.
We shall use a different formulation which has been presented in [17] and which offers some advantages over the conventional method. To set up our numerical model, define on the original silicon and silicon dioxide range, i.e. on $\Omega = \Omega_1 \cup \Omega_2$ of Fig. 3, a function of relative silicon concentration,

$$\eta(x, t) = \frac{C_{Si}(x, t)}{C_{SiO}}.$$  \hspace{1cm} (1)

$C_{Si}(x, t)$ is the silicon concentration at time $t$ and spatial coordinate $x$. $C_{SiO}$ is the concentration of silicon atoms in pure crystal. Hence $\eta$ is 1 in pure silicon, 0 in pure oxide and between 0 and 1 around the interface $\Gamma_1$. (The oxidant diffusion in nitride can be neglected, therefore this part of the domain will only be considered for the structural deformation.)

As the time scale for the diffusion process is much smaller than that of the oxide growth, the diffusion steps can be considered stationary on a sequence of fixed domains. The oxidant diffusion in $\Omega_1 \cup \Omega_2$ can be described by

$$\nabla(D(\eta(x, t)) \nabla C(x, t)) = k(\eta(x, t)) C(x, t),$$  \hspace{1cm} (2)

for every fixed time $t$, with boundary conditions

$$C = C^* \text{ on } \Gamma_2$$  \hspace{1cm} (3)

and

$$\frac{\partial C}{\partial n} = 0 \text{ on } \Gamma_3 \cup \Gamma_4.$$  \hspace{1cm} (4)

$k$ is the strength of a spatial sink. $kC$ defines how many particles of oxygen per unit volume react in a unit time interval to oxide.

Now let $N_i$ be the number of oxidant molecules in a unit volume of silicon dioxide and $\lambda$ be the volume expansion factor ($=2.2$) for the reaction from silicon to oxide. Obviously, the chemical reaction in the interface range changes the relative silicon concentration $\eta$, i.e. in a time interval $t, t + \Delta t$ and a test volume $\Delta V$ (where we assume the oxidant concentration $C$ to be constant), $k(\eta(t)) C(x, t) \Delta V \Delta t$ particles of oxygen react with $k(\eta(t)) C(x, t) \Delta V \Delta t / (N_i \lambda)$ unit volumes of silicon to form

$$V^{\text{new}}_{\text{SiO}_2} = k(\eta(t)) C(x, t) \Delta V \Delta t / N_i$$  \hspace{1cm} (5)
unit volumes of silicon dioxide. Letting $\Delta t \to 0$, we conclude that

$$ \frac{\partial \eta}{\partial t} = - \frac{1}{\lambda} k(\eta(t)) C(x, t) / N_1. $$

(6)

The additional volume per unit volume silicon after oxidation is given by

$$ V^{\text{new}} = \frac{\lambda - 1}{\lambda} \Delta t \Delta V k(x, \eta(t)) C(x, t) / N_1, $$

(7)

which produces prestrains in the control volume and thus loads the displacement problem.

The incremental prestrains $d\varepsilon_0$ are coupled to incremental stresses $d\sigma$ in an incremental elastic deformation problem (which is valid for temperatures up to about 1000 °C) by the material law

$$ d\sigma = M(\eta)(d\varepsilon - d\varepsilon_0), $$

(8)

with an elasticity tensor $M$ depending on the relative silicon density $\eta$.

In our numerical algorithm, the overall domain of computation is meshed into triangular finite elements. Linear or quadratic shape functions are used to model the diffusion problem (2)–(4) and the mechanical problem in the classical manner. As all material properties, i.e. the diffusion and reaction coefficients in (2) and also the elasticity and shear modulus defining $M$ vary with $\eta$ in space and time, numerical integration has to be used to compute element matrices. Therefore, $\eta$ has to be defined for every integration point, coupling the diffusion part to the deformation part of the local oxidation algorithm.

The moving interface between silicon and oxide is only implicitly defined in our approach. As a choice for an equivalent sharp interface line [17], the contour line for $\eta = 0.5$ is used. Thus, the interface is obtained by postprocessing the primary finite element results. The following advantages are due to this implicit interface definition.

- The equivalent interface line (i.e. the contour line for $\eta = 0.5$) does not need to be tracked by element boundaries; therefore it is not necessary to remesh the oxide domain after every time step.
- In several advanced oxidation processes, initially separated oxide ranges grow together during oxidation. This topological change in the oxide domain imposes severe problems to any interface-tracking algorithm, yet it does not present difficulties in the implicit formulation.
- As silicon and oxide are modelled jointly, a stress distribution can be computed not only in oxide but also in silicon (which often is technologically more significant).

The structure of the numerical algorithm is outlined as follows:

1. Define an initial domain $\Omega$ and a silicon distribution $\eta$, thus introducing the dioxide, silicon and interface range. (The discretization of nitride is straightforward and is not described explicitly.)
2. Define a finite element mesh, initialize time $t = 0$.
3. Initialize material coefficients $D$, $k$, elasticity modulus $E$ and shear modulus $\nu$ at all integration points.
4. Solve the quasistationary diffusion problem (2)–(4).
(5) Compute the reacting amount of oxidant and silicon at each integration point from (5).
(6) Update the silicon distribution function $\eta$ at all integration points according to (6).
(7) Compute incremental prestrains at all integration points from the reacting volume.
(8) Solve the mechanical deformation problem.
(9) Update the nodal coordinates.
(10) Update the total stresses at the integration points.
(11) Set $t = t + \Delta t$ and goto 3.

Fig. 4. SEM-photograph and finite element mesh: initial domain.

Fig. 5. SEM-photograph and finite element mesh: domain after oxidation.
An example with a comparison of the simulated oxide body and the physical experiment is shown in Figs. 4 and 5. On the left-hand side of Fig. 4, the scanner electron photograph of the simulated structure on the right-hand side is presented; Fig. 5 shows the same structure after oxidation. For a more detailed description see [18] or [19]. It is important to note that the numerical interface width, i.e. the range where \( \eta \) varies from 0 to 1 does not need to coincide with the physical interface width, if one is only interested in the shape of the oxide body (see [17]). Yet it is also an interesting question to study the micromechanical behaviour of the interface layer itself. If this is the objective of the simulation, it is of course necessary to resolve the physical interface width by a proper discretization. As a typical size of the total domain of simulation is 1–2 \( \mu \)m and the interface width is roughly 5–10 nm, the relative width of the reactive layer is only 1/100 to 1/400.

Reconsidering the structure of (2), we can observe that the global solution behaviour is, due to the diffusive nature of the equation, very smooth (except for some singular points at the boundary). Therefore the \( p \)-version is well-suited to accurately model the global behaviour. Only at the interface will there be a very steep descent of the oxide concentration \( C \) making it necessary to strongly refine the finite element mesh. The domain decomposition approach that is presented in the next sections is related to that of Ewing and co-workers [10–12] which was originally developed for oil reservoir simulation. We extend their work and combine it with \( h-p \) version concepts and with the hierarchical finite element approach [20].

3. An \( h-p \) domain decomposition

The \( h-p \) domain decomposition is outlined on the model equation

\[
\nabla(D(x)\nabla C) = k(x)C \quad \text{on } \Omega,
\]

with

\[
C = C_0 \quad \text{on } \Gamma^c,
\]

\[
\frac{\partial C}{\partial n} = 0 \quad \text{on } \Gamma^p,
\]

\( \Omega \) is a polygonal domain and it is assumed for simplicity that \( D \) and \( k \) are piecewise constant.

Consider now the non-standard finite element mesh as shown in Fig. 6 with a refined zone around the reactive layer.

We construct a domain-decomposition method which

— uses \( p \)-version finite elements on the coarse mesh,
— uses an overlay of \( p \)-version and \( h \)-version approximation on the fine mesh,
— guarantees \( C^0 \)-continuity across the coarse/fine-mesh boundary,
— can be implemented into a \( p \)-version code without major modifications in program and data structure.

The basic idea is demonstrated for the one-dimensional model of Fig. 7(a–c). Figure 7(a) shows a coarse mesh with three elements and a \( p \)-approximation function. In Fig. 7(b) an
**E. Rank, Adaptive remeshing and h-p domain decomposition**

Fig. 6. Geometrically incompatible finite element mesh.

Fig. 7. One-dimensional approximation.

$h$-version approximation on the second 'coarse' element with zero boundary values at the coarse mesh/fine mesh interface is plotted. Finally, Fig. 7(c) shows that the hierarchical sum of the coarse and fine mesh functions is continuous by construction.

The two-dimensional approximation space can be constructed as follows. Let $\Omega_2 \subset \Omega_1$ be open, bounded domains and let $\Gamma_2$ be the boundary of $\Omega_2$ (Fig. 8). Let $T_1 = \{ t_{1,i}, i \in I_1 \}$ be a regular triangulation of $\Omega_1$ which is chosen so that there is a subset $I_{2,i} \subset I_1$ with

$$\Omega_2 = \bigcup \{ t_{1,i}, i \in I_{2,i} \}. \tag{10}$$

![Fig. 8. Two-dimensional domain decomposition.](image)
Let $T_2 = \{ t_{2,i}, i \in I_2 \}$ be a regular triangulation of $\Omega_2$ which is chosen so that for every $t_{2,i} \in T_2$ there is exactly one $j \in I_{2,1}$ with

$$i_{2,i} \subset t_{1,j},$$

(11)
i.e. elements of $T_2$ do not intersect edges of the triangulation $T_1$.

On $\Omega_2$, we now choose $S_{h,\Omega_2}$ as a ‘standard’ finite element space over $T_2$ and a subspace

$$S_{h,\Omega_2}^0 := \{ v \in S_{h,\Omega_2} \mid v = 0 \text{ on } \Gamma_2 \}.$$  

(12)

Typically, we use for $S_{h,\Omega_2}$ an approximation of low order finite elements, for example $p = 1$ or $p = 2$. Similarly, we define a $p$-version approximation space $S_{p,\Omega_1}$ over the triangulation $T_1$ with a subspace

$$S_{p,\Omega_1}^0 = S_{p,\Omega_1} \setminus S_{h,\Omega_2}^0.$$  

(13)

The global approximation space is now defined as

$$S_{h,p} = S_{h,\Omega_2}^0 \oplus S_{p,\Omega_1}^0,$$

(14)
i.e. every function in $S_{h,p}$ can be represented by the sum of a (global) $p$-version function and a (local) $h$-version function.

Now let $\{ N_i^h \mid i \in I_h \}$ be the set of shape functions of the $h$-approximation on the fine mesh, $\{ N_i^p \mid i \in I_p \}$ the set of shape functions of the $p$-approximation on the coarse mesh. Then the system-matrix in a finite element formulation for (9) is defined as

$$K := (k_{ij})_{i,j \in I_h \cup I_p},$$  

(15)

with

$$k_{ij} := B(N_i, N_j) := \int_{\Omega} D(x)\nabla N_i \nabla N_j - k(x)N_i N_j \, d\Omega.$$  

(16)

Ordering all indices so that $h$-version shape functions get the highest numbers, the structure of the global system matrix is as follows:

$$K = \begin{bmatrix} K_{pp} & K_{hp} \\ K_{hp}^t & K_{hh} \end{bmatrix}.$$  

(17)

$K_{pp}$ and $K_{hh}$ can be computed in standard manner, $K_{hp}$ is the coupling matrix of fine mesh/low order modes to coarse mesh/high order modes.

There are various possibilities to solve the resulting system of linear equations. In [11], an iterative precondtioning technique is presented and analysed, which could be used for solving the system. As the goal of the present work was to implement the multilevel method into an existing $p$-version code with only minor modifications in program and data structure, a different method was chosen.

As usual, the system matrix (15) is computed by assembling element matrices, i.e. by
performing integration in (16) element-wise. In our formulation, there are three groups of elements. The first are \( p \)-version elements in \( \Omega_1 \setminus \Omega_2 \), i.e. those with no hierarchical refinement. The corresponding element matrices are computed with a standard Gaussian integration rule. The second group are \( p \)-version elements in \( \Omega_2 \). Their element matrices are integrated by a composed integration rule, first splitting the elements into triangles that are geometrically identical to the overlaid fine mesh and then integrating over each of the small triangles. This integration procedure allows us to compute the \( p \)-version element matrices together with the coupling terms of \( K_{hp} \) simultaneously with fine mesh element matrices composing \( K_{hh} \).

To be precise, consider Fig. 9 with a fine mesh element \( t_h \) and its coarse mesh ‘father’-element \( t_p \). Let \( N^h := (N_i^h, \ldots, N_k^h) \) be the \( h \)-version shape functions on \( t_h \) and \( N^p := (N_i^p, \ldots, N_k^p) \), the \( p \)-version shape functions on \( t_p \). Defining a differential operator \( S \) as

\[
S := \begin{bmatrix}
\frac{\partial}{\partial x} \\
\frac{\partial}{\partial y} \\
1
\end{bmatrix},
\]

(18)

corresponding to the bilinear form in (16), we obtain a ‘\( B \)-matrix’ at a point \( x \) in \( t_h \) as

\[
B(x) := S \cdot N := S \cdot (N^h, N^p).
\]

Using numerical integration over \( t_h \) with \( g \) integration points \( x_1, \ldots, x_g \) and weights \( w_1, \ldots, w_g \), we can compute a hierarchical element matrix

\[
K^h := \sum_{i=1}^g w_i \cdot B(x_i) \begin{bmatrix}
D(x_i) & 0 & 0 \\
0 & D(x_i) & 0 \\
0 & 0 & k(x_i)
\end{bmatrix} B(x_i).
\]

(20)

\( K^h \) contains the element matrix part of \( t_h \) in \( K_{hh} \), the coupling terms corresponding to \( t_h \) in \( K_{hp} \) and the part of the element matrix of \( t_p \) corresponding to the small element \( t_h \).

The only difference between (20) and a standard element matrix computation is that the vector of the shape functions \( N^h \) has to be augmented by the \( p \)-version shape functions \( N^p \). In

Fig. 9. Coarse mesh and fine mesh elements.
a computer program this means that at each integration point in \( t^h \), in addition to the \( t^h \)-shape functions and derivatives, those of \( t^p \) have to be computed. To do this, it is only necessary to compute local coordinates with respect to the father element and to 'call' a subroutine for the \( p \)-version functions at that point.

It remains to be explained how conditions (12) and (13) can be satisfied in a computer program. Condition (12) is completely straightforward. All \( h \)-version modes on \( \Gamma_2 \) have to be eliminated to guarantee continuity of the solution. This is done by imposing Dirichlet boundary conditions for every \( h \)-version mode on \( \Gamma_2 \). To implement condition (13), it is necessary to eliminate all modes of the \( p \)-approximation which are linear combinations of \( h \)-version modes. If the \( h \)-version uses only linear approximation (as implemented in our computer code), these are exactly all basic modes at nodes strictly interior to \( \Omega \). These nodes can easily be found only from geometric data by preprocessing element lists.

4. Mesh generation and data structures

In this section, it is shown how standard mesh generation programs and standard data structures can be used with only minor modifications for an implementation of the proposed method. Let us start with a mesh description data structure which has been derived from that of the mesh generator TRIGEN described in [21]. Basic elements are (see Fig. 10) nodes, edges, regions and triangles or quadrilaterals. A region is a simply connected, bounded domain, defined by a sequence of consecutive edges \( \text{BOUNDARY}_1 \) to \( \text{BOUNDARY}_N \). Of course it is possible to define not only 1 but \( k \) regions, thus making it possible to describe different material types or multiply connected domains. This data structure is sufficient to describe the coarse mesh for the \( p \)-version approximation. Our mesh generator is able to generate fully automatically pure triangular or pure quadrilateral meshes according to a mesh density function which can either be user-defined or derived from a posteriori error estimation. For more details, see [22].

```
NODE = X = Y =

EDGE = NODE.1 = NODE.2 =

REGION = MAT =
BOUNDARY.1 =

BOUNDARY.N =

TRIANGLE = REGION = N1 = N2 = N3 =
QUADRILATERAL = REGION = N1 = N2 = N3 = N4 =
```

Fig. 10. Basic data structure.
REGION = MAT = IS_DAUGHTER_OF =

TRIANGLE
QUADRILATERAL = IS_FATHER = YES/NO
IS_FATHER = YES/NO

Fig. 11. Extended data structure.

After a first analysis with a $p$-version approximation, refinement zones are marked by a posteriori criteria. For our model problem (9), the gradient of the corresponding relative silicon density (1), (2) is used indicating the range of the interface between silicon and oxide.

In a second data structure, each triangle or quadrilateral in the refinement zone of the coarse mesh defines a new region for the second mesh generation. This correspondence of father-elements to daughter-regions guarantees automatically that elements in $T_2$ do not intersect edges of the coarse mesh elements of $T_1$. The extended data structure (Fig. 11) is thus derived from the basic data-structure by adding one flag (IS_FATHER) and one pointer (IS_DAUGHTER_OF). Note that the same data structure can be used for multiply refined zones, i.e. by an overlay of more than two meshes.

5. Numerical examples

For the numerical examples, the domain with a basic triangulation as shown in Fig. 12 with $k/D = 1 / 100$ (nm$^{-1}$) and a diameter of the structure of 1000 nm was used. The diffusion and reaction parameters correspond to an interface width of about 10 nm. It is well known (see [6, 23]) that low order approximation leads to severe oscillations of the solution due to a

Fig. 12. Basic triangulation for the test problem.
violation of the discrete maximum principle. This can be seen in Fig. 13(a,b). Contour lines of
the solution for $p = 2$ on the mesh of Fig. 12 are plotted in Fig. 13(a). Figure 13(b) shows the
same results in bird's eye view.

Figure 14 shows a locally refined mesh and Fig. 15 the finite element solution for $p = 2$.
Oscillations in the silicon range disappear, yet the interface width is still overestimated by a
factor of more than 2. As expected, a 'standard' $h$-$p$ version with $p = 4$ and the refined mesh
of Fig. 14 leads to excellent results as shown in Fig. 16(a,b).

The $h$-$p$ domain decomposition starts on the coarse mesh of Fig. 12. In Fig. 17 the
refinement zone is marked; the composed mesh after the second triangulation has already
been shown in Fig. 6. Finally, in Fig. 18(a,b) contour lines and bird's eye view are plotted.
Slight oscillations indicated by contour lines in Fig. 18(a) are small enough not to be visible in
the 3D-plot of Fig. 18(b). The resolution of the interface is as good as that in the pure $h$-$p$
version approximation.
Fig. 15. $p=2$ on refined mesh: contour lines and bird's eye view.

Fig. 16. $p=4$ on refined mesh: contour lines and bird's eye view.

Fig. 17. Refinement zone for $hp$-$d$ version.
6. Conclusions

A combination of an $h$-$p$ version approximation and a domain decomposition method has been presented and it has been shown that this approach can be implemented into existing $p$-version codes with only minor changes in program and data structure. The method is particularly interesting for problems with interior curves of singular behaviour. Various extensions of the proposed method are possible. A parallel implementation interpreting each triangle of the coarse mesh in the refined region as a substructure promises to be very attractive. Another interesting investigation could be the application of the method to transient problems with moving fronts, as one could take advantage of the unchanged basic mesh, only locally remeshing at the proceeding layer of singular behaviour.

References